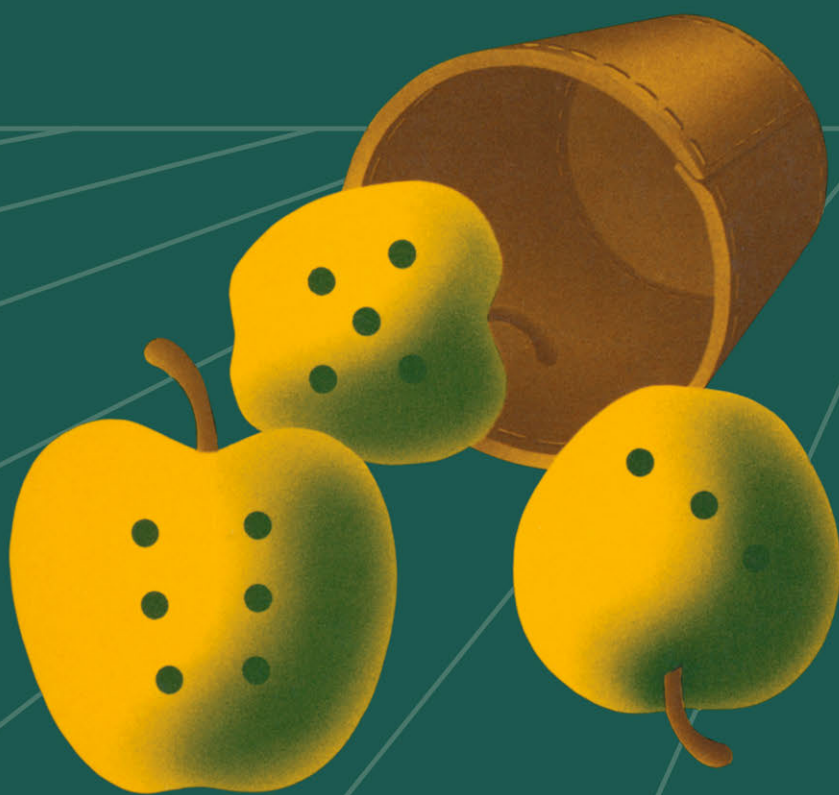


Franklin-Koltnow-Finkel

SPIELPROGRAMME FÜR DEN Apple IIe

Anleitungen – Techniken – Unterprogramme



Springer Fachmedien Wiesbaden GmbH

Howard Franklin / Joanne Koltnow / LeRoy Finkel

**Spielprogramme
für den APPLE IIe**

Howard Franklin

Joanne Koltnow

LeRoy Finkel

Spielprogramme für den APPLE IIe

Spiele sowie Anleitungen, Techniken
und Unterprogramme
für die Eigenentwicklung von Spielen

Übersetzt von Leo Schaaf



Springer Fachmedien Wiesbaden GmbH

Autorisierte Übersetzung der amerikanischen Originalausgabe
H. Franklin/J. Koltnow/L. R. Finkel
Golden Delicious Games for the APPLE Computer
Copyright © 1982 by John Wiley & Sons, Inc., New York
Aus dem Amerikanischen von Leo Schaaf

Das in diesem Buch enthaltene Programm-Material ist mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Der Autor übernimmt infolgedessen keine Verantwortung und wird keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieses Programm-Materials oder Teilen davon entsteht.

Alle Rechte an der deutschen Ausgabe vorbehalten
© der deutschen Ausgabe Springer Fachmedien Wiesbaden 1984
Ursprünglich erschienen bei Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig 1984.
Die Vervielfältigung und Übertragung einzelner Textabschnitte, Zeichnungen oder Bilder, auch für Zwecke der Unterrichtsgestaltung, gestattet das Urheberrecht nur, wenn sie mit dem Verlag vorher vereinbart wurden. Im Einzelfall muß über die Zahlung einer Gebühr für die Nutzung fremden geistigen Eigentums entschieden werden. Das gilt für die Vervielfältigung durch alle Verfahren einschließlich Speicherung und jede Übertragung auf Papier, Transparente, Filme, Bänder, Platten und andere Medien.

Umschlaggestaltung: Peter Lenz, Wiesbaden
Satz: Vieweg, Braunschweig

ISBN 978-3-528-04270-7 ISBN 978-3-663-13935-5 (eBook)
DOI 10.1007/978-3-663-13935-5

Vorwort der Autoren

Das Buch „Spielprogramme für den APPLE IIe“ enthält neue Spiele sowie Verbesserungen bekannter Gesellschaftsspiele und Vorschläge bzw. Anregungen zum Programmieren weiterer Spiele. Es wurde speziell für diejenigen Leser geschrieben, die bereits ausreichend mit der Programmiersprache BASIC vertraut sind und bessere, abwechslungsreiche Programme schreiben möchten. Das Buch wendet sich an Eltern, Lehrer, Schüler und Studenten oder einfach an den Computer-Hobbyisten.

Wir stellen sorgfältig konstruierte Programme zur Verfügung. Damit können Klänge oder Farbmuster erzeugt, Daten direkt nach Eingabe ausgewählt und bewertet oder bestimmte Tasten deaktiviert werden. Die Programme eignen sich als Zusätze für existierende Programme und als Grundbausteine für neue Programme.

Wir haben Unterprogramme in eigenständige Spielprogramme eingebaut. Sowohl die Unterprogramme als auch die Spielprogramme können als Beispiele guten Programmierstils angesehen werden. Mit ihnen wollen wir unsere Überzeugung von der Bedeutung benutzerfreundlicher Computerprogramme ausdrücken. Geben Sie die Programme so ein, wie sie in diesem Buch abgedruckt sind. Speichern Sie diese unter den Namen, die wir angegeben haben. Dies gibt uns die Möglichkeit bei der Konstruktion größerer Programme auf die vorhergehenden Arbeiten Bezug zu nehmen. Außerdem erspart es Ihnen die Zeit des wiederholten Eintippens. Lesen Sie die Kapitel der Reihe nach! Lassen Sie kein Kapitel aus, und Sie werden sehen, daß Sie dadurch Gewinn und Freude steigern.

Zur Benutzung des Buches benötigen Sie einen APPLE II Computer mit APPLESOFT (FP) BASIC Programmiersprache. Einige unserer Programme sind sehr kurz und benötigen nicht mehr als 16K Speicherkapazität, doch die meisten Programme brauchen 32K. Zusätzlich benötigen Sie ein Diskettenlaufwerk zum Speichern der Programme und Unterprogramme. Mit einem Kassettenrekorder ist das Speichern der Programme sehr viel komplizierter. Für die Verwendung eines Diskettenlaufwerks spielt es keine Rolle, ob das Betriebssystem DOS 3.2. oder 3.3. benutzt wird (DOS = Disk Operating System).

Einige Bemerkungen zu Computerspielen

Heutzutage sind nur wenige Computerspiele wirklich neu. Die meisten sind zuerst für große Computersysteme geschrieben worden (vor 15 bis 20 Jahren die einzigen erhältlichen Computer). Diese frühen Spiele wurden an Fernschreiber-Terminals gespielt und basierten deshalb auf der Verwendung von Texten. Wenn Sie heute Textzeilen aus dem Bildschirm verschwinden sehen, dann erinnern Sie sich an folgendes: Die Papierrollen des Fernschreibers hatten keine Begrenzung, wie dies beim Bildschirm der Fall ist. Die Programmierer wußten, daß die Spieler jederzeit bei fehlenden Anweisungen die Rolle zurückdrehen konnten, um diese zu finden.

Wir erwähnen dies aus zwei Gründen:

Erstens ist es gut zu wissen, warum die Dinge sich so entwickelt haben, wie sie heute sind. Computerspiele haben eine geschichtliche Entwicklung, und – so oft wie möglich – werden wir auch die Ursprünge der vorgestellten Spiele aufzeigen.

Zweitens haben viele heutige Spiele noch Nachteile, da sie lediglich Mikrocomputer-Adaptionen der früheren, textorientierten Fernschreiberspiele sind. Wir denken z.B. an das Verschwinden von Textzeilen aus dem Bildschirmbereich. Auch werden die Vorteile des Mikrocomputers oft nicht voll ausgenutzt. Im Verlauf dieses Buches werden wir einige Verfahren zum Verbessern bereits existierender Spiele durch die speziellen Eigenschaften des APPLE vorschlagen.

Noch ein Wort an die Leser

Die Computer-Programme von „Spiele für den APPLE-Computer“ werden auf nicht kopiergeschützten Disketten geliefert. Es gibt keine mechanischen oder elektronischen Tricks, um das Kopieren zu verhindern. Die Programme sind jedoch urheberrechtlich geschützt. Wir fühlen uns moralisch verpflichtet, Software in einer Form anzubieten, die der Leser auch abändern kann. Der Leser kann zwei Dinge tun: die Funktion unserer Spiele erforschen, indem er einzelne Teile unserer Programme ausprobiert, und eigene Spiele entwickeln. Außerdem hat er ein Anrecht auf eine ausreichende Zahl von Sicherheitskopien, so daß er nicht zu befürchten braucht, daß maschinelle Fehlfunktionen oder persönliche Fehler die einzige Kopie des Programms zerstören.

An dieser Stelle möchten wir die persönliche Bitte an den Leser richten, unsere Programme nicht zu kopieren, um sie unter Bekannten und Freunden zu vertreiben. Mit dem Verzicht auf einen Kopierschutz hoffen wir, den Software-Herstellern wie auch den Herausgebern zu zeigen, daß Programme auch ohne Kopierschutz mit finanziellem Gewinn verbreitet werden können.

Wenn es keinen finanziellen Gewinn gibt, können die Hersteller die Computer-Software künftig nicht mehr zu vernünftigen Preisen anbieten.

Nehmen Sie bitte Rücksicht auf unsere Bemühungen und machen Sie nur Kopien für Ihren eigenen Gebrauch!

Howard Franklin
Joanne Koltnow
LeRoy Finkel

Vorwort des Übersetzers

Das vorliegende Buch enthält neben vielen Programmen, die das Erstellen von Computerspielen wesentlich erleichtern, eine Reihe von fertig ausgearbeiteten Spielen. Als Vorlage für die Computerversionen dieser Spiele dienten den Autoren bekannte amerikanische Kinderspiele, die nicht immer im Deutschen eine Entsprechung besitzen. Deshalb wurde größtenteils der Originalname des Spiels beibehalten (z.B. MATCH, BLOCKOUT). Die Übertragung der Programme ins Deutsche geschah lediglich durch Übersetzung der Dialogformen und der REM-Anweisungen. Der Ablauf der Programme blieb ungeändert. Trotzdem kann für vollständige Fehlerfreiheit aller Programmlistings keine Gewährleistung übernommen werden.

L. Schaaf

Dezember 1983

Inhaltsverzeichnis

Vorwort der Autoren	V
Vorwort des Übersetzers	VII
Kapitel 1 Melodien und Klangeffekte	1
Kapitel 2 Die niederauflösende Grafik	16
Kapitel 3 Bilder in LO-RES-Grafik	30
Kapitel 4 Hochauflösende Grafik	49
Kapitel 5 Programme zur Dateneingabe	56
Kapitel 6 Wortspiele	72
Kapitel 7 Weitere Spiele	93
Anhang A RENUMBER/MERGE-Hilfsprogramm	109
Anhang B Randbemerkungen zum Programmieren der Spiele	111
Anhang C Hinweise zu den Programmlistings	121
Anhang D Das Entwickeln von Programmen	123
Sachwortverzeichnis	125
VIII	

Kapitel 1

Melodien und Klangeffekte

Dieses Kapitel führt in grundlegende Möglichkeiten zur Klangerzeugung des APPLE-Computers ein. Außerdem enthält es Unterprogramme zur Erzeugung von Klangeffekten, die Sie zu bereits existierenden Programmen hinzufügen können. Mit diesen Unterprogrammen als Grundlage können Sie neue Programme in allen gewünschten Variationen erstellen.

Der Ablauf der Programme auf dem Bildschirm kann in diesem Buch nicht gezeigt werden. Es ist nicht möglich, den Ablauf eines Programms darzustellen, das Klänge produziert; es ist schon schwierig, einen Programmablauf zu demonstrieren, bei dem sich farbige Bilder verändern. Wir finden es jedoch angebracht, die von uns geschaffenen Wahlmöglichkeiten zu erläutern, mit denen ein Programm auf den Spieler eingeht. Deshalb enthalten die Kapitel Erörterungen über spezielle Dialoge zwischen Spieler und Programm.

Der Apple-Computer erzeugt Töne durch das sehr schnelle Ein- und Ausschalten eines elektrischen Schalters im Innern des Computers. Indem man eine lange Folge dieser winzigen Schaltvorgänge in einen Lautsprecher leitet, entsteht ein Ton. Ändert man die Anzahl der Schaltvorgänge pro Sekunde, so ändert sich auch der Ton. Alle verschiedenen Klänge, die der APPLE-Computer erzeugt, kommen von diesen Schaltvorgängen. Sie werden durch einen Lautsprecher wiedergegeben. Zum Beispiel erfordert ein Klang, dessen Tonhöhe etwa dem Kammerton a entspricht, 440 Schaltvorgänge in der Sekunde.

Die BELL-Funktion

Wenn Sie die Control-G Taste drücken, wird Ihr APPLE-Computer einen Ton (Beep) von sich geben, der manchmal auch als Bell bezeichnet wird. (Die CTRL-Taste bleibt gedrückt während man die G-Taste betätigt.) Wiederholen Sie dies einige Male und achten Sie auf den Ton. Der Control-G-Befehl bewirkt diesen Ton durch die Benutzung eines im Computer gespeicherten Unterprogramms, das eine bestimmte Folge von Schaltvorgängen erzeugt.

Haben Sie bemerkt, daß die G-Taste auch mit BELL bezeichnet ist? Die Bezeichnung ist ein Überbleibsel aus jenen Tagen, als in Fernschreibgeräten noch solche Bells eingebaut waren und durch Eintippen von Control-G betätigt wurden. Heutzutage klingt "BELL" gewöhnlich nicht wie eine Klingel, doch die meisten Terminals besitzen irgendwelche akustischen Signale.

Angenommen Sie wollen den Beep benutzen, um in einem Spiel eine richtige Antwort oder einen siegreichen Schritt anzuzeigen. Sie können Control-G als Teil des Programms in einen PRINT-Befehl mit aufnehmen. Control-Zeichen erscheinen jedoch nicht im Programmlisting; sie sind zwar in Ihren PRINT-Befehlen vorhanden, werden aber im Programmlisting vermißt.

```
PRINT "DAS IST RICHTIG!!"  
(nicht sichtbares Control-G)
```

Das ist zwar nicht schlecht, doch es kann ärgerlich und verwirrend sein, wenn Sie Ihre Listings betrachten. Glücklicherweise gibt es eine Alternative. CHR\$ (7) ist der ASCII-Code für Control-G, und diese Zeichen erscheinen im Programmlisting. Benutzen Sie CHR\$ (7) in einem PRINT-Befehl wie folgt:

```
PRINT "DAS IST RICHTIG!!"; CHR$ (7)  
PRINT CHR$ (7); "DAS"; CHR$ (7); "IST"; CHR$ (7); "RICHTIG!!";  
CHR$ (7)
```

Sie stellen fest, daß das Eintippen von CHR\$ (7) sehr schnell mühsam werden kann. Daneben kann man ja auch vergessen, welche Zahl zu benutzen ist. Wenn wir für CHR\$ (7) eine Textvariable definieren, sind die PRINT-Befehle leichter einzugeben. Als Variable für Beep werden wir BL\$ benutzen:

```
10 BL$ = CHR$ (7)  
20 PRINT BL$; "DAS"; BL$; "IST"; BL$; "RICHTIG!!"; BL$
```

Pausen

Beim Ausprobieren dieser Programmzeilen mit dem APPLE werden Sie feststellen, daß Wörter und Töne beinahe gleichzeitig erscheinen. Der PRINT-Befehl wird so schnell ausgeführt, daß nur schwer zu sagen ist, ob die Töne auf die Wörter folgen. Wenn Sie einen Ton mehrere Male spielen wollen, verschwimmen die Töne ineinander. (Der Befehl PRINT BL\$; BL\$; BL\$ hört sich wie ein einziger Ton anstatt wie drei Töne an.)

Mit dem SPEED-Kommando wird die Geschwindigkeit kontrolliert, in der durch einen PRINT-Befehl Zeichen auf dem Bildschirm ausgegeben werden. SPEED = 255 ist die schnellste, SPEED = 0 die langsamste Geschwindigkeit. Wenn keine Geschwindigkeit festgelegt ist, wird die vorgewählte Geschwindigkeit (255) benutzt. Hier zeigen wir einen interessanten Weg, das SPEED-Kommando zur Kontrolle der Pausen zwischen den Tönen zu benutzen.

```

10 REM ...TOENE PAUSE...
11 :
12 REM FUEGE EINE PAUSE ZWISCHEN 2 TOENEN EIN UNTER BENUTZUNG VON 'SPEED'
13 :
100 BL$ = CHR$ (7)
110 SPEED= 0
120 PRINT BL$;BL$;BL$
130 SPEED= 255

```

Lassen Sie das Programm mit SPEED = 0 laufen. Verändern Sie die Geschwindigkeit in Zeile 110 zu 50, 100, 150, etc., und starten Sie das Programm erneut.

Spiele mit Tönen

Einfache Programme, die nur einen Ton benutzen, sind sehr leicht zu entwerfen. Gesetzt den Fall, Lehrer oder Eltern möchten diesen Ton in einem Programm dazu benutzen, kleinen Kindern das Zählen beizubringen. So fragt z.B. ein solches Programm den Spieler nach einer Zahl zwischen 1 und 10. Dann zeigt das Programm die Zählfolge bis zu dieser Zahl und hebt dabei jede Zahl durch einen Ton hervor. Wenn ein Kind z.B. die Zahl 5 eingibt, so erzeugt das Programm einen Ton und zeigt die Zahl 1 an, der nächste Ton und die 2 erscheint, der nächste Ton und die 3 erscheint, usw. Das spielende Kind zählt die Töne, während es die Zahlen, die auf dem Bildschirm erscheinen, beobachtet.

Hier ist das Programm:

```

10 REM ...EINGABE TOENE...
11 :
12 REM ERZUEGE TONANZAHL LAUT EINGABE
13 :
100 L = 1: REM MINIMALE TONANZAHL
110 H = 5: REM MAXIMALE TONANZAHL
200 TEXT : HOME
210 PRINT "BITTE NENNE EINE ZAHL VON ";L;" BIS ";H;: INPUT ": ";N
220 IF N < L OR N > H THEN 200
230 PRINT
300 PRINT "ZAEHLE DIE TOENE....": PRINT : PRINT
310 FOR J = 1 TO 1000: NEXT : REM PAUSE VOR DEM ERSTEN TON
320 FOR J = 1 TO N
330 SPEED= 0
340 PRINT " ";: REM LASSE ZEIT VERSTREICHEN (2 * LEERTASTE)
350 SPEED= 255
360 PRINT J;
370 PRINT CHR$ (7);
390 NEXT
500 PRINT : VTAB 18
510 PRINT "DRUECKE RETURN UND VERSUCHE NOCH EINMAL. "
515 PRINT "DRUECKE LEERTASTE FUER PROGRAMMENDE...."
520 GET Z$
530 IF Z$ = CHR$ (32) THEN END : REM ABFRAGE VON ESC
540 GOTO 200

```

Speichern Sie dieses Programm als EINGABE TOENE ab, und starten Sie einen Probelauf.

Frage 1: Schauen Sie sich das Programmlisting an. Wie wird die maximale Anzahl von Tönen auf 20 verändert?

Dieses Programm ist etwas ungewöhnlich, da es dem Spieler laufend Anweisungen erteilt. Bei den meisten Lehrprogrammen und Spielen erhält jedoch der Computer die Befehle. Tatsächlich testet bzw. rechnet der Computer häufiger, als daß er lehrt. Beim Schreiben von Programmen ist darauf zu achten, wer die Anweisungen während eines Spiels gibt. Gewöhnlich macht es dem Spieler mehr Spaß, wenn er die Kontrolle über den Computer ausübt.

Eine herkömmliche Variante des Spielens mit Tönen und Zahlen ist folgende: Das Programm wählt eine Zahl, gibt mehrere Töne von sich und fragt den Benutzer nach der Anzahl der Töne. Dieses neue Programm testet die Fähigkeit des Spielers im Zählen der Töne. Es spornt auch an, wenn die Antwort nahe beim richtigen Ergebnis liegt (siehe Zeile 120 und 430 im Programm weiter unten).

Anstatt das ganze Programm einzugeben, können wir auch das vorherige Programm EINGABE TOENE wie folgt abändern:

Die Zeilen 220, 230 und 360 werden gelöscht und anschließend folgende Zeilen eingefügt. (Einige der folgenden Zeilen beinhalten lediglich kleine Veränderungen der bereits vorhandenen Zeilen.)

```
10 REM ...ZAEHLE-EINGABE-TOENE...
11 :
100 L = 4: REM MINIMALE ANZAHL VON TOENEN
110 H = 16: REM MAXIMALE ANZAHL VON TOENEN
120 C = 1: REM WIE NAHE AM RICHTIGEN ERGEBNIS?
210 N = INT ((H - L + 1) * RND (1)) + L: REM # OF BEEPS
380 HTAB 1
400 INPUT "NUN, WIE VIELE TOENE WAREN DIES? ";G
410 PRINT
420 IF G = N THEN PRINT "DAS IST RICHTIG !!!": GOTO 500
430 IF ABS (G - N) < = C THEN PRINT "FAST,ABER NICHT GANZ GENAU.
.. ES SIND.. ";N;".": GOTO 500
440 PRINT "SPIEL NOCHMAL... ES WAREN ";N;"."
```

Speichern Sie das Programm als ZAEHLE TOENE und starten Sie es.

Frage 2: Wie ändern Sie die Variable für die Ausgabe eines aufmunternden Textes auf 3?

Weitere Töne

Die Nutzung des bisherigen Tones eröffnet interessante Möglichkeiten zum Programmieren, doch kann der APPLE ebenso viele andere Töne produzieren. Wie bereits erwähnt, bestimmt die Anzahl der Schaltvorgänge pro Sekunde (Frequenz) die Höhe des computererzeugten Tones.

Als erstes geben Sie das folgende Programm ein und speichern es als NEXT-DATA-EINHEIT ab. Dieses Programm erlaubt der READ DATA-Anweisung, bei einer beliebigen Zeilennummer zu beginnen. Diese Einheit kann in allen Programmen dazu benutzt werden, den DATA-Zeiger auf eine bestimmte Zeilennummer zurückzusetzen. (siehe Anhang B für zusätzliche Erläuterungen.)

```

10 REM    ...NEXTDATA EINHEIT...
11 :
12 REM    BEGINNE READ IN BELIEBIGER ZEILE...
13 :
18991 :
18992 :
18993 REM    ** NAECHSTE DATEN IN ZEILE Z **
18994 REM    EINGANGSVARIABLE: Z  ZEILENNUMMER
18999 :
19000 IF YR% THEN 19200: REM    UEBERPRUEFE,OB NEXTDATA PROGRAMM GEL
      ADEN IST
19010 YR% = 770: REM    NAECHSTE DATA ADRESSE
19097 :
19098 REM    NEXT DATA PROGRAMM GESCHRIEBEN IN MASCHINENSPRACHE
19099 :
19100 POKE 770,173: POKE 771,0: POKE 772,3: POKE 773,133: POKE 774,
      80: POKE 775,173: POKE 776,1: POKE 777,3
19110 POKE 778,133: POKE 779,81: POKE 780,32: POKE 781,26: POKE 782
      ,214: POKE 783,165: POKE 784,155: POKE 785,24
19120 POKE 786,105: POKE 787,4: POKE 788,133: POKE 789,125: POKE 79
      0,165: POKE 791,156: POKE 792,105: POKE 793,0
19130 POKE 794,133: POKE 795,126: POKE 796,96
19200 Z% = Z / 256: POKE YR% - 2,Z - 256 * Z%: POKE YR% - 1,Z%: REM
      LINE #
19210 CALL YR%
19220 RETURN

```

Speichern Sie dieses Programm als NEXTDATA-EINHEIT ab.

Dies ist das erste von sogenannten "Black-box"-Programmen, die wir Ihnen zur Verfügung stellen. Der Begriff "Black-box" wird allgemein für etwas verwendet, dessen Auswirkung leicht erkennbar ist, dessen genaue Funktionsweise aber verborgen bleibt. Unsere "Black-box"-Programme sind in Maschinensprache geschrieben. Führen wir diese Programme ein, so erläutern wir zwar ihre Funktion, behandeln aber nicht den Programmaufbau in allen Einzelheiten. Ihre komplizierte Konstruktion würde das Ziel dieses Buches übersteigen. Für interessierte Leser verweisen wir auf Anhang B. Die meisten von Ihnen werden die Programme auch ohne nähere Erläuterung benutzen, um sich das Programmieren zu erleichtern.

Fügen Sie das folgende Listing an NEXTDATA-EINHEIT an:

```

10 REM ...KLANG EINHEIT-NEXTDATA EINHEIT..
11 :
12 REM KLANG PROGRAMM ZUM SPIELEN VON BUCHSTABENFOLGEN UND ERZEUG
    EN VON KLANGEFFekten
13 :
12991 :
12992 :
12993 REM ** WIEDERGABE VON TONHOEHE FUER FESGESETZTE LAENGE **
12994 REM EINGANGSVARIABLE: WP TONHOEHE #
12995 REM (WP=0 UND PROGRAMM NICHT GELADEN = NUR INITI
    ALISIERUNG)
12996 REM WD TONDAUER
12999 :
13000 IF WR% THEN 13200: REM UEBERPRUEFE OB KLANG PROGRAMM SCHON
    GELADEN
13010 WR% = 800:WP% = 799:WD% = 797: REM TON,TONHOEHE, TONDAUER A
    DRESSEN
13020 Z = 13100: GOSUB 19000: REM SETZE READ-DATA-ZEIGER
13050 Z = WR%: REM LADE KLANG PROGRAMM
13060 READ Z1: IF Z1 > = 0 THEN POKE Z,Z1:Z = Z + 1: GOTO 13060
13070 IF WP = 0 THEN RETURN : REM SPERRE NUR FUER INITIALISIERUN
    G
13097 :
13098 REM KLANG PROGRAMM GESCHRIEBEN IN MASCHINENSPRACHE
13099 :
13100 DATA 172,31,3,185,73,3,141,31,3,160,0,238,29,3,238,30,3,174,
    31,3,173,48,192
13110 DATA 136,208,10,206,29,3,208,5,206,30,3,240,5,202,240,234,20
    8,238,96
13117 :
13118 REM TONHOEHEN
13119 :
13120 DATA 255,242,228,215,203,192,181,171
13130 DATA 161,152,143,135,127,120,113,107
13140 DATA 101,95,90,85,80,75,71,67
13150 DATA 63,59
13190 DATA -1: REM ZEICHEN,UM DATEN EINLESEN ZU BEENDEN
13200 Z% = WD / 256: POKE WD%,WD - 256 * Z%: POKE WD% + 1,Z%: REM
    TONDAUER
13210 POKE WP%,WP: REM TONHOEHE #
13220 CALL WR%
13230 RETURN
13292 :
13293 REM * SPIELE BUCHSTABENKOMBINATION *
13294 REM EINGANGSVARIABLE: Z% TEXT
13295 REM WD TONDAUER
13299 :
13300 IF LEN (Z%) = 0 THEN RETURN : REM LEERER TEXT
13310 FOR W = 1 TO LEN (Z%)
13320 WP = ASC ( MID% (Z%,W,1)) - 64: REM NAECHSTER BUCHSTABE
13330 IF WP > = 1 AND WP < = 26 THEN GOSUB 13000: REM SPIELE,
    WENN BEDINGUNG ERFUELLT
13340 NEXT
13350 RETURN
13382 :
13383 REM * KLANG EFFEKTE *
13384 REM EINGANGSVARIABLEN: W1 LAENGE DER TOENE (>=0)

```

```

13385 REM                                W2 SCHRITTWEITE ZWISCHEN TONEN (>0
)
13386 REM                                (W2=0 UND PROGRAMM NICHT GELADEN = N
UR INITIALISIERUNG)
13387 REM                                W3 ERSTER TON (0/255)
13388 REM                                W4 ANZAHL DER TONE EINES ZYKLUS
13389 REM                                W5 1=ZURUECK;-1=VOR;0=VOR UND ZURUECK
13390 REM                                W6 PAUSE ZWISCHEN DER WIEDERHOLUNG
DER ZYKLEN
13391 REM                                W7 ANZAHL DER WIEDERHOLUNGEN
13399 :
13400 IF WE% THEN 13500: REM  UEBERPRUEFE OB EFFEKTE EINGELESSEN
13410 IF WR% = 0 THEN WP = 0: GOSUB 13000: REM  LESE KLANG PROGRAM
M EIN,WENN NOTWENDIG
13420 WE% = 809: REM  ADRESSEN FUER KLANGEFFEKTE
13430 IF W2 = 0 THEN RETURN : REM  SPERRE NUR FUER INITIALISIERU
NG
13500 WH% = W1 / 256:WL% = W1 - 256 * WH%: REM  TONHOEHE ZWEI BYTES
13510 IF W2 < = 0 THEN W2 = 1: REM  MACHE W2 GUELTIG
13520 IF W3 < 0 THEN W3 = 0: REM  MACHE W3 GUELTIG
13530 FOR Z = 1 TO W7: REM  ANZAHL DER WIEDERHOLUNGEN
13540 Z% = W3 + W2 * W4: IF W5 < 0 THEN 13600: REM  SPERRE NUR VORW
AERTS
13550 FOR Z1 = W3 TO Z% STEP W2: REM  ZYKLUS RUECKWAERTS
13560 IF Z1 < = 255 THEN POKE WP%,Z1: POKE WD%,WL%: POKE WD% + 1,
WH%: CALL WE%: REM  NAECHSTER TON IST GUELTIG
13570 NEXT
13600 IF W5 > 0 THEN 13650: REM  SPERRE NUR ZURUECK
13610 FOR Z1 = Z% TO W3 STEP - W2: REM  ZYKLUS VORWAERTS
13620 IF Z1 < = 255 THEN POKE WP%,Z1: POKE WD%,WL%: POKE WD% + 1,
WH%: CALL WE%: REM  NAECHSTER TON IST GUELTIG
13630 NEXT
13650 FOR Z1 = 1 TO W6: NEXT : REM  PAUSE ZWISCHEN 2 ZYKLEN
13660 NEXT
13670 RETURN

```

Geben Sie dieses Listing ein und speichern es als **KLANG EINHEIT** ab.

Dies ist eine Sammlung von Unterprogrammen, die in anderen Programmen benutzt werden können; lediglich für sich alleine sind die Unterprogramme nicht lauffähig!

Musiknoten

Die folgenden Änderungen an **KLANG EINHEIT** ergeben ein Programm, bei dem die Tasten mit den Ziffern 1 bis 8 den Noten der Tonleiter entsprechen: Löschen der Zeilen 13120 bis 13150.

```

10 REM  ...TASTEN1/8-KLANG EINHEIT...
11 :
12 REM  PROGRAMM ZUM SPIELEN MIT DEN TASTEN 1 BIS 8
13 :
100 GOSUB 13000: REM  INITIALISIERE KLANG EINHEIT
200 TEXT : HOME
210 PRINT "'SPIELE' EINEN TON,BENUTZE DIE          TASTEN 1 BIS 8"

```

```

220 PRINT
230 PRINT "DRUECKE RETURN ZUM BEENDEN DES TONES.."
240 PRINT "DRUECKE LEERTASTE FUER PROGRAMME..."
250 PRINT
300 GET Z$
310 IF Z$ = CHR$ (13) THEN 500: REM RETURN
320 IF Z$ = CHR$ (32) THEN END : REM LEERTASTE
330 IF Z$ < "1" OR Z$ > "8" THEN 300: REM IGNORIERE ANDERE TASTEN

340 PRINT Z$;
350 WP = ASC (Z$) - 48: REM WANDLE UM AUF 1/8
360 WD = 50: REM TONDAUER
370 GOSUB 13000
380 GOTO 300
500 PRINT : VTAB 18
510 PRINT "DRUECKE RETURN UND VERSUCH'S NOCH EINMAL.....";
520 GET Z$
530 IF Z$ = CHR$ (32) THEN END : REM LEERTASTE
540 GOTO 200

```

Speichern Sie dieses Programm als TASTEN 1/8 ab.

Starten Sie es und spielen Sie einfache Melodien durch Betätigen der Tasten 1 bis 8. Während des Spielens erscheinen die Ziffern der betätigten Tasten auf dem Bildschirm. Sie können sie abschreiben und sich so Tonfolgen festhalten, die Ihnen gefallen. Wie bei unseren anderen Programmen endet der Ton nach Drücken der RETURN-Taste. Mit der Leertaste wird das Programm gestoppt.

Das Programm KLANG EINHEIT läßt den Computer bei einer bestimmten Anzahl verschiedener Frequenzen Töne erzeugen. Es arbeitet ähnlich wie das interne Programm, das durch Eingabe von Control-G aktiviert wird. Diesmal wurden Frequenzen gewählt, die ungefähr der Tonleiter entsprechen. Es wurden die Ziffern 1 bis 8 benutzt, um die Tonleiter zu spielen. Die Zuordnung der Tonleiter zu den Tasten 1 bis 8 ist willkürlich und wird bestimmt durch die DATA-Anweisungen in den Zeilen 13120 bis 13150.

Spielen Sie die folgende Sequenz von Noten und fügen bei jedem Sternchen eine Pause ein: 123455*66665*66665*444433*55551.

Frage 3: Wie lautet die Melodie?

Durch Änderung der DATA-Anweisungen in den Zeilen 13120 bis 13150 können wir uns zusätzliche Frequenzen schaffen. Den neuen Tönen sind nun in alphabetischer Reihenfolge die Buchstaben der Tastatur zugeordnet. Das folgende Programm liefert daher eine chromatische Tonleiter mit 26 Tönen.

Hier sind die Befehle, die die Funktion der Tasten 1 bis 8 auf die Tasten A bis Z übertragen. Machen Sie im Programm TASTEN 1/8 folgende Änderungen:

```

10 REM ...TASTEN A/Z - TASTEN 1/8
11 :
12 REM PROGRAMM ZUM "SPIELEN" AUF DEN TASTEN A BIS Z IN ALPHABE
TISCHER REIHENFOLGE
13 :
210 PRINT "'SPIELE' EINE MELODIE; BENUTZE DIE TASTEN A BIS Z"

330 IF Z$ < "A" OR Z$ > "Z" THEN 300: REM IGNORIERE ANDERE TASTEN

350 WP = ASC (Z$) - 64: REM WANDLE UM AUF A/Z
13120 DATA 255,242,228,215,203,192,181,171
13130 DATA 161,152,143,135,127,120,113,107
13140 DATA 101,95,90,85,80,75,71,67
13150 DATA 63,59

```

Speichern Sie dieses Programm als TASTEN A/Z ab. Starten Sie es und spielen Sie mit den Tasten A bis Z auf der Tastatur. Achten Sie auf die Töne. Wenn Sie eine hübsche Melodie gefunden haben, notieren Sie die Buchstaben auf dem Bildschirm, damit Sie das Lied später wieder spielen können.

Musikalische Botschaft

Nun stellen wir Ihnen ein Programm vor, das Melodien mittels Information aus dem Datenspeicher spielt und nicht direkt mit der Tastatur. Es erlaubt eine bestimmte Folge von Tönen von der Tastatur in den Speicher zu lesen. Nach Betätigen der RETURN-Taste spielt das Programm die Tonfolge; es ist nicht mehr notwendig, für jeden Ton eine Taste zu drücken.

Ändern Sie TASTEN A/Z wie folgt:

Streichen Sie die Zeilen 300 bis 380 und fügen die folgenden Zeilen hinzu:

```

10 REM ...MUSIK BOTSCHAFT TASTEN A/Z
11 :
12 REM PROGRAMM ZUR EINGABE UND ZUM SPIELEN VON TEXTEN(ALPHABETIS
CH)
13 :
230 PRINT "GIB 'MELODIE' EIN , DANN DRUECKE RETURN ZUM SPIELEN"
240 PRINT
250 INPUT "MELODIE: ";Z$
300 WD = 50: REM TONDAUER
310 GOSUB 13300: REM SPIELE TEXT

```

Speichern Sie dieses Programm als MUSIK BOTSCHAFT ab. Starten Sie es und benutzen Sie einige der Melodien, die Sie bereits notiert haben. Wie wäre es, wenn Sie Ihren Namen eingeben und hören, wie der Computer dies spielt? Wie klingen die Namen Ihrer Stadt oder Ihres Landes? Ein leicht variiertes Programm könnte nach Ihrem Namen fragen und diesen mehrmals spielen, vielleicht in veränderter Reihenfolge.

Frage 4: Wie wird das Programm geändert, damit eine Melodie dreimal anstatt nur einmal gespielt wird?

Ändern Sie MUSIK BOTSCHAFT durch Hinzufügen folgender Zeilen:

```
10 REM ...VOR UND ZURUECK-MUSIK BOTSCHAFT...
11 :
12 REM GIB EINEN TEXT EIN, DANN SPIELE VOR UND ZURUECK
13 :
110 BF = 1: REM WIE OFT VOR UND ZURUECK?
250 INPUT "TUNE: ";F$
260 B$ = ""
270 IF LEN (F$) THEN FOR J = 1 TO LEN (F$):B$ = B$ + MID$ (F$, LEN
(F$) + 1 - J,1): NEXT J: REM TEXT RUECKWAERTS
290 FOR J = 1 TO BF
305 Z$ = F$: REM VORWAERTS
320 Z$ = B$: GOSUB 13300: REM ZURUECK
330 NEXT
```

Speichern Sie dieses Programm als VOR/ZURUECK und starten Sie es. Tippen Sie auf den Tasten A bis Z Ihre Melodie ein und hören Sie sich das Ergebnis an. Wir haben nun ein Programm, das eine Folge von Buchstaben zuerst in der eingegebenen Reihenfolge abspielt und dann nochmals in der umgekehrten Richtung. Nachdem Sie ein paar Wörter und Formulierungen ausprobiert haben, versuchen Sie einige 'Palindrome' einzutippen, um zu hören, wie diese klingen. (Ein 'Palindrom' ist eine Folge von Buchstaben, die in beiden Richtungen gelesen denselben Wortlaut ergibt. Zwei Beispiele hierfür sind: „Eine treue Familie bei Lima feuerte nie“ oder „Neuer Dienst mag Amtsneid reuen“.)

Frage 5: Alle Töne haben die gleiche Länge. Welche Zeile muß geändert werden, um die Tondauer zu verlängern?

Frage 6: Wie lautet die Variable, die angibt, wie oft die Buchstabenfolge vorwärts und rückwärts zu spielen ist?

Piano

Eine weitere Änderung des Tastenfeldes beinhaltet das folgende PIANO-Programm. Diesmal wird jeder Buchstabe des Tastenfeldes mit einer Note gleichgesetzt. Anstatt einer alphabetischen Ordnung entsprechen nun die Tasten von links nach rechts Tönen mit steigender Frequenz.

Um das Programm PIANO zu erstellen, machen Sie bitte folgende Änderungen im Programm TASTEN A/Z (nicht bei VOR/ZURUECK, obwohl diese Änderungen das Tastenfeld bei VOR/ZURUECK auch beeinflussen):

```
10 REM ...PIANO-TASTEN A/Z
11 :
12 REM "PIANO" VERWENDUNG DER TASTEN A/Z
13 :
210 PRINT "'PIANO' UNTER VERWENDUNG DER TASTEN A-Z"
13120 DATA 171,203,228,152,90,143,135,127
13130 DATA 67,120,113,107,181,192,63,59
13140 DATA 101,85,161,80,71,215,95,242
13150 DATA 75,255
```

Speichern Sie dieses Programm als PIANO ab. Alles was verändert wurde, war wiederum nur die Zuordnung der Tonhöhen zu den einzelnen Tasten in den Zeilen 13120 bis 13150. Mit diesem Programm kann auf der Tastatur des Computers wie auf einem Piano gespielt werden. (Mit Ausnahme der Tatsache, daß alle Töne die gleiche Länge haben und nicht mehrere Tasten gleichzeitig gespielt werden können.) Sicherlich fällt es mit dieser Anordnung der Töne jetzt leichter, Melodien zu spielen.

Elektrische Orgel

Eine Randbedingung aller bisherigen Programme ist, daß die Töne eine festgelegte Dauer haben. Wir können die Länge der Töne variieren, aber wir sind nicht in der Lage, die Dauer jedes einzelnen Tones unabhängig von den anderen Tönen zu verändern. Das folgende Programm macht von einer gänzlich anderen Möglichkeit der Tondauerbestimmung Gebrauch. Ein Ton dauert so lange an, bis eine neue Taste gedrückt wird — dieser Effekt imitiert also eine elektrische Orgel.

Modifizieren Sie das Programm PIANO wie folgt:

```

10 REM ...ORGEL-PIANO...
11 :
12 REM ELEKTRISCHE ORGEL
13 :
100 GOSUB 13700: REM INITIALISIERE ORGEL PROGRAMM
210 PRINT "'ORGEL' VERWENDET DIE TASTEN A BIS Z"
370 GOSUB 13700
13692 :
13693 REM * ORGEL *
13694 REM EINGANGSVARIABLE: WP TONHOEHE
13695 REM (WP=0 UND PROGRAMM NICHT GELADEN = NUR INITI
ALISIERUNG)
13699 :
13700 IF WS% THEN 13900: REM UEBERPRUEFE,OB ORGEL-PROGRAMM GELADE
N
13710 WS% = 882: REM ORGEL ADRESSEN
13720 IF WR% = 0 THEN W = WP:WP = 0: GOSUB 13000:WP = W: REM LADE
KLANG PROGRAMM(SPEICHERE TONHOEHE)
13730 Z = 13800: GOSUB 19000: REM SETZE READ-DATA-ZEIGER
13750 Z = WS%: REM LADE ORGEL PROGRAMM
13760 READ Z1: IF Z1 > = 0 THEN POKE Z,Z1:Z = Z + 1: GOTO 13760
13770 IF WP = 0 THEN RETURN : REM SPERRE NUR FUER INITIALISIERUN
G
13797 :
13798 REM ORGEL PROGRAMM GESCHRIEBEN IN MASCHINENSPRACHE
13799 :
13800 DATA 172,31,3,185,73,3,141,31,3,173,0,192,48,14,174,31,3,173
,48,192
13810 DATA 136,208,0,202,240,239,208,248,96
13890 DATA -1: REM ZEICHEN ZUM BEENDEN DES DATENEINLESSENS
13900 POKE WP%,WP: REM TONHOEHE
13910 CALL WS%
13920 RETURN

```

Speichern Sie dieses Programm als ORGEL ab. Spielen Sie damit, um zu sehen, wie es sich vom Programm PIANO unterscheidet.

Klang-Effekte

Zum Schluß möchten wir das sehr leistungsstarke Programm KLANG EFFEKTE vorstellen. Da dieses Programm viele Möglichkeiten eröffnet, zeigen wir Ihnen einen systematischen Weg auf, es kennenzulernen.

Eine Vielfalt von Klangeffekten wird möglich, wenn wir die folgenden Zeilen zum Programm KLANG EINHEIT hinzufügen.

```
10 REM ...KLANG EFFEKTE-KLANG EINHEIT..
11 :
12 REM KLANG EFFEKT ENTWICKLUNG
13 :
100 GOSUB 13400: REM INITIALISIERE KLANG EFFEKTE PROGRAMM
210 W1 = 0
220 W2 = 1
230 W3 = 0
240 W4 = 10
250 W5 = 1
260 W6 = 200
270 W7 = 4
300 TEXT : HOME
310 PRINT : PRINT "LAENGE JEDES TONES : ";W1
320 PRINT : PRINT "SCHRITTWEITE ZWISCHEN ZWEI TOENEN :";W2
330 PRINT : PRINT "ERSTER TON : ";W3
340 PRINT : PRINT "ANZAHL DER TOENE IN EINEM ZYKLUS: ";W4
350 PRINT : PRINT "1=ABWAERTS; -1=AUFWAERTS; 0=AUF-UND ABWAERTS";W5
360 PRINT : PRINT "PAUSE ZWISCHEN ZWEI WIEDERHOLUNGEN: ";W6
370 PRINT : PRINT "ANZAHL DER WIEDERHOLUNGEN: ";W7
500 PRINT : VTAB 18
510 PRINT "DRUECKE RETURN UND HOEREN SIE...";
520 GET Z$
530 IF Z$ = CHR$ (32) THEN END : REM LEERTASTE
540 GOSUB 13400
600 PRINT : PRINT
610 PRINT "DRUECKE RETURN UND PROBIEREN SIE NEUE VARIABLEN AUS....";
620 GET Z$
630 IF Z$ = CHR$ (32) THEN END : REM LEERTASTE
700 TEXT : HOME
710 PRINT "GEBEN SIE FUER JEDEN PARAMETER EINEN WERT EIN ODER DRUECKE
N SIE RETURN,UM DEN ALTEN WERT BEIZUBEHALTEN."
810 PRINT : PRINT "ALTE DAUER EINES JEDEN TONES: ";W1;" NEW: ";; INPUT
Z$: IF LEN (Z$) THEN W1 = VAL (Z$)
820 PRINT : PRINT "ALTE SCHRITTWEITE ZWISCHEN 2 TOENEN: ";W2;" NEW: ";
: INPUT Z$: IF LEN (Z$) THEN W2 = VAL (Z$)
830 PRINT : PRINT "ALTER START TON : ";W3;" NEW: ";; INPUT Z$: IF LEN
(Z$) THEN W3 = VAL (Z$)
840 PRINT : PRINT "ALTE ANZAHL VON TOENEN IM ZYKLUS: ";W4;" NEW: ";; INPUT
Z$: IF LEN (Z$) THEN W4 = VAL (Z$)
850 PRINT : PRINT "ALTER AUF/ABWAERTS PARAMETER ";W5;" NEW: ";; INPUT Z
$: IF LEN (Z$) THEN W5 = VAL (Z$)
860 PRINT : PRINT "ALTE PAUSE ZWISCHEN 2 ZYKLEN ";W6;" NEW: ";; INPUT
Z$: IF LEN (Z$) THEN W6 = VAL (Z$)
870 PRINT : PRINT "ALTE ANZAHL VON WIEDERHOLUNGEN: ";W7;" NEW: ";; INPUT
Z$: IF LEN (Z$) THEN W7 = VAL (Z$)
890 GOTO 300
```


Speichern Sie dieses Programm unter dem Namen KLANG EFFEKTE ab. Lassen Sie es einige Male ablaufen und lesen Sie danach an dieser Stelle weiter.

Das Programm gibt die Zahlenwerte jedes Parameters mittels Bildschirm an und erzeugt nach Drücken der RETURN-Taste den dadurch bestimmten Klangeffekt. Nun fragt es nach den Änderungen der Parameter, wobei jeder Wert einzeln abgefragt wird. (Durch einfaches Drücken der RETURN-Taste wird der aktuelle Wert beibehalten.).

Die Vielfalt von Klangeffekten, die durch dieses Programm erzeugt werden, ist außerordentlich groß. Obwohl es verlockend ist, mit jedem Durchlaufen des Programms alle Parameter zu variieren, wird es zum Kennenlernen nützlicher sein, nur ein oder zwei Parameter gleichzeitig zu ändern.

Haben Sie Töne gefunden, die Ihnen gefallen, variieren Sie die benutzten Zahlenwerte. Auf diese Weise sehen Sie, ob der Klang weiter verfeinert werden kann. Notieren Sie sich die Zahlenwerte, damit das Programm mit diesen speziellen Werten auch für zukünftige eigene Programmentwicklungen zu benutzen ist.

Schauen Sie zuerst, wie sich der Klang ändert, wenn der Anfangston verändert wird. Die möglichen Töne in einem Zyklus reichen von 0 (hoher Ton) bis 255 (tiefer Ton). Wir starten mit 0, also dem höchsten Ton. Versuchen Sie einige Anfangstöne, die tiefer liegen. Ursprünglich haben wir die Anzahl der Töne in einem Zyklus auf zehn festgesetzt; versuchen Sie, diese zu verkleinern. Haben Sie bemerkt, daß bei kürzer werdenden Zyklen der Klang einem Knacken ähnelt? Die Schrittweite gibt die Anzahl von Tönen zwischen zwei verschiedenen Noten an. Bei größerer Schrittweite wird der resultierende Klangeindruck weniger fließend sein.

Nun möchten wir die Anzahl der Wiederholungen eines Zyklus und die Länge der Pause zwischen zwei Zyklen verändern. Keine dieser Änderungen hat große Auswirkungen auf den Klang. Dagegen wird die Änderung des AUF/ABWARTS-Parameters den Höreindruck bedeutend verändern.

Es gibt folgende Möglichkeiten: 1 (nur aufwärts), - 1 (nur abwärts) und 0 (auf- und abwärts).

An dieser Stelle wird der Leser schon etwas vertrauter mit den bisher erläuterten Programmteilen sein. Nachdem das Programm einige Zeit benutzt wurde, ist es leicht die Art der Klangeindrücke vorherzusagen, die durch die unterschiedlichen Werte der Variablen verursacht werden.

Bis hierhin haben Sie sicher eine kleine Sammlung von Zahlenkombinationen aufgeschrieben, die Klänge und Töne nach Ihrem Geschmack erzeugen. Beim Einbau dieses Programms in eigene Programme werden die Werte den entsprechenden Parametern zugeordnet, um so die gewünschten Klangeffekte zu erzeugen.

Zusammenfassung

Im ersten Kapitel haben wir gezeigt, wie die BELL-Taste und der SPEED-Befehl genutzt werden können. Wir haben außerdem ein eigenständiges Programm zur Simulation eines Pianos erhalten und ein weiteres, das eine Orgel simuliert.

Das nützlichste Programm dieses Kapitels ist KLANG EINHEIT. Dieser Baustein erlaubt es, Musiktöne aller Art zu erzeugen und exotische Klangeffekte zu produzieren. Wir werden KLANG EINHEIT in einigen weiteren Programmen in diesem Buch verwenden.

Unterprogramm zur Klangerzeugung: Zusammenfassende Hinweise

Wir haben gezeigt, wie die verschiedenen Klangmöglichkeiten des APPLE beeinflusst werden können. Nun wollen wir demonstrieren, wie eigene Programme des Lesers mit Tönen und Musik ausgestattet werden können.

Die Variablenbezeichnungen W, X, Y, und Z werden in unseren Unterprogramm-Einheiten benutzt und sollten daher in eigenen Programmen nicht mehr verwendet werden. (mit Ausnahme der Kommunikation mit unseren Unterprogrammen). Weiter sollten die Programme keine Zeilennummern zwischen 10000 und 50000 enthalten, denn dort sind unsere Programm-listings angesiedelt.

Töne und Melodien: Zusammenfassung

Um Musik mit den Tasten der Buchstaben A bis Z zu erzeugen, müssen wir folgendes wissen:

Beginn des Unterprogramms:	13300
Eingangsvariablen:	Z\$ Textvariable
	WDTondauer

Ein Musikprogramm kann zum Beispiel so aussehen:

```
1220 Z$ = "LUSTIGE APPLE SPIELE"  
1230 WD = 100  
1240 GOSUB 13300  
1250 :  
1260 REM FORTSETZUNG DES PROGRAMMS.
```

So einfach ist das Ganze!

Klangeffekte: Zusammenfassung

Um Klangeffekte zu erzeugen, können Sie bis zu sieben Variablen festlegen oder ihren vorbestimmten Wert benutzen.

Beginn des Unterprogramms:	13400
Eingangsvariablen:	
W1: Länge der Töne:	= 0
W2: Schrittweite zwischen den Tönen:	= 0

W3: Anfangston: 0 bis 255

W4: Anzahl der Töne in einem Zyklus

W5: 1 = Tonfolge abwärts, - 1 = Tonfolge aufwärts, 0 = Tonfolge auf- und abwärts

W6: Pause zwischen Wiederholungen

W7: Anzahl der Wiederholungen

Ein Programm zur Erzeugung von Klangeffekten kann wie folgt geschrieben werden:

```
1300 W1 = 4: W2 = 1: W3 = 50: W4 = 20: W5 = 0: W6 = 200: W7 = 4
```

```
1310 GOSUB 13400
```

```
1320 :
```

```
1330 REM FORTSETZUNG DES EIGENEN PROGRAMMS
```

Lösungen zu Kapitel 1

Frage 1: 110 H = 20

Frage 2: 120 C = 3

Frage 3: Alle meine Entchen ...

Frage 4: 305 FOR J = 1 TO 3
314 NEXT

Frage 5: Zeile 300

Frage 6: Ändere den Wert von BF in Zeile 110

Kapitel 2

Die niederauflösende Grafik

In diesem Kapitel werden die Grundprinzipien der niederauflösenden Grafik (im folgenden LO-RES-Grafik genannt) und die Programmierung einiger Farbeffekte erläutert. Wir zeigen, auf welche Weise Farbpunkte auf dem Bildschirm dargestellt werden können. Anschließend erweitern wir unsere Überlegungen auf farbige Linien, Rechtecke, Begrenzungen und zeigen Programme, die den ganzen Bildschirm mit Farbe ausfüllen.

Das Kapitel soll uns mit dem Gebrauch der LO-RES-Grafik intensiv vertraut machen, damit die Möglichkeiten von LO-RES in eigenen Programmen verwendet werden können. Auch wenn Sie die speziellen Programme dieses Buches nicht benutzen, können Sie die hier gemachten Überlegungen anwenden und eigene Programme mit Farbeffekten erstellen. (In Kapitel 3 werden wir sehen, wie Bilder oder Muster von LO-RES-Farbpunkten erstellt und verändert werden.)

Der APPLE-Computer hat sechzehn Farben zur Verfügung, die auf dem Bildschirm bzw. Farb-Monitor gezeigt werden. Durch Gebrauch der niederauflösenden Grafik können wir diese Farben verwenden. Geringe Auflösung bedeutet, daß Details nur begrenzt darstellbar sind. Der kleinste adressierbare (d.h. veränderbare) Punkt ist etwa halb so groß wie ein auf dem Schirm ausgedrucktes Textzeichen. Im Gegensatz dazu können mit der hochauflösenden Grafik sehr viel kleinere Punkte dargestellt werden; dies ergibt Bilder mit besserer Detailwiedergabe, doch leider sind bei der hochauflösenden Grafik nur sechs Farben verfügbar. Wir werden auf die hochauflösende Grafik in Kapitel 4 näher eingehen.

Es gibt zwei Formen der LO-RES-Grafik. Die erste hat vierzig Grafikzeilen und ein vier Zeilen großes Textfenster. Die andere erlaubt es, den ganzen Bildschirm (48 Zeilen) mit Grafik auszufüllen. Die erste Form werden wir des öfteren benutzen, denn sie gibt uns die Möglichkeit, Textanweisungen zusammen mit der Grafik auf dem Bildschirm erscheinen zu lassen.

Text und Grafik können nur durch Verwendung des vier Zeilen großen Textfensters gemischt werden. Später werden wir zeigen, wie mittels LO-RES-Grafik Blockbuchstaben zum Schreiben von Wörtern oder Zahlen erstellt werden.

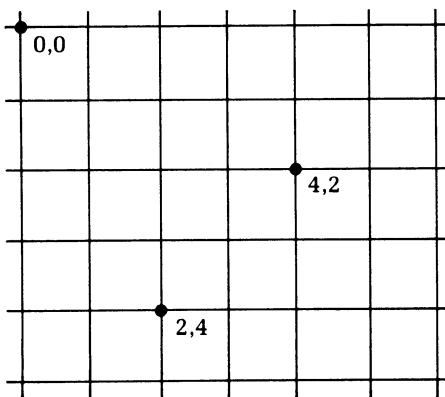
Die Farbgrafik des APPLE-Computers

Zur Programmierung von Farbeffekten in LO-RES-Grafik werden nur fünf Befehle benötigt: GR, COLOR, PLOT, HLIN und VLIN. Mit GR schaltet der APPLE in die mit dem Textfenster versehene Form der LO-RES-Grafik. Der Bildschirm wird gelöscht und erscheint völlig schwarz. (Die zweite Form der LO-RES-Grafik wird weiter unten behandelt.)

Mit COLOR wird eine bestimmte Farbe gewählt. Sie wird so lange verwendet, bis sie mit einem anderen COLOR-Befehl geändert wird. So ergibt z. B. der Befehl COLOR = 4 die Farbe dunkelgrün. Die Zuordnung der Farbe ist aus der nachfolgenden Tabelle ersichtlich:

0 SCHWARZ	8 BRAUN
1 MAGENTAROT	9 ORANGE
2 DUNKELBLAU	10 GRAU 2
3 VIOLETT	11 ROSA
4 DUNKELGRÜN	12 HELLGRÜN
5 GRAU 1	13 GELB
6 BLAU	14 AQUAMARINBLAU
7 HELLBLAU	15 WEISS

Der Befehl PLOT weist den APPLE an, einen Farbpunkt an einer bestimmten Stelle zu zeichnen. Beim Arbeiten mit Textfenster ist der Bildschirm in vierzig mal vierzig Punkte unterteilt. Die Punkte sind von 0 bis 39 numeriert. Jeder Punkt wird durch ein Zahlenpaar beschrieben, wobei die H-Koordinate (horizontal) an erster Stelle, die V-Koordinate (vertikal) an zweiter Stelle angegeben wird. Der Punkt 0,0 befindet sich in der linken, oberen Ecke des Bildschirms. Der Befehl PLOT 3,9 bewirkt, daß ein Punkt in der dritten Spalte von links und der neunten Zeile von oben gezeichnet wird.



Mit dem Befehl HLIN werden zwei Punkte einer bestimmten Zeile horizontal verbunden. Mit VLIN werden zwei Punkte einer bestimmten Spalte vertikal verbunden. So bewirkt z. B. HLIN 3,23 AT 9 das Zeichnen einer horizontalen Linie in Zeile neun, von der dritten bis zur dreiundzwanzigsten Spalte.

Zusammengefaßt: Ein farbiger Punkt wird auf folgende Weise erzeugt:

- Umschalten auf LO-RES-Grafik,
- Wahl einer Farbe,
- Eingabe des PLOT-Befehls.

Mit weiteren PLOT-Befehlen werden andere Punkte derselben Farbe gezeichnet. Die Befehle HLIN und VLIN bewirken das Zeichnen horizontaler und vertikaler Linien. Durch Verwenden des COLOR-Befehls wird die Farbe vor dem Zeichnen des nächsten Punktes verändert.

Probieren Sie folgendes Programm aus:

```
100 GR: COLOR = 4
110 HLIN 3,23 AT 9
120 VLIN 3,25 AT 23
130 END
```

Farbige Punkte

Hier ist ein einfaches Programm, das an zufällig gewählten Stellen des Bildschirms farbige Punkte ausdruckt.

```
10 REM ...FARBPUNKTE...
11 :
100 GR : HOME
200 PRINT
210 PRINT "DRUECKE EINE BELIEBIGE TASTE ZUM          BEENDEN DES PROGRAMMS
    ....";
300 H = INT (40 * RND (1))
400 V = INT (40 * RND (1))
500 COLOR= INT (16 * RND (1))
600 PLOT H,V
700 P = 200
710 FOR Z = 1 TO P: NEXT : REM PAUSE
800 IF PEEK ( - 16384) < 128 THEN 300: REM KEINE TASTE GEDRUECKT
810 GET Z$: REM LOESCHE TASTENDRUCK
```

Die Bildschirmkoordinaten zum Ausdruck der Punkte werden in den Zeilen 300 (horizontal) und 400 (vertikal) festgelegt. Für jede Koordinate wird eine Zahl zwischen 0 und 39 per Zufall ermittelt. Für die Farben wird eine Zufallszahl zwischen 0 und 15 gewählt. Damit stehen alle möglichen Farben zur Verfügung. Der PLOT-Befehl wird in Zeile 600 erteilt.

Die Gleichungen in den Zeilen 300 und 400 können verallgemeinert werden. So ist es möglich, eine zufällige Zahl zwischen zwei beliebigen Zahlen A und B auszuwählen. Die allgemeine Anweisung lautet:

$$\text{LET R} = \text{INT} ((\text{B} - \text{A} + 1) * \text{RND}(1)) + \text{A}$$

Wichtig in diesem Programm ist schließlich das Programmieren einer Pause in Zeile 700. Das Verändern des Zahlenwertes der Variable ändert die Zeitspanne zwischen dem Ausdrucken zweier Punkte.

Geben Sie dieses Programm ein und speichern Sie es unter dem Namen FARBPUNKTE. Starten Sie das FARBPUNKTE-Programm. Es kann beendet werden durch das Betätigen einer beliebigen Taste. Dabei bleibt der Rechner jedoch auf LO-RES-Grafik geschaltet. In dieser Form stehen nur vier Textzeilen am unteren Bildschirmrand zur Verfügung.

Um den ganzen Bildschirmbereich wieder für Text verwenden zu können, geben Sie das Kommando TEXT ein. Auf dem Bildschirm erscheinen eine Vielzahl von schwarzen und weißen Zeichen, die in schneller Folge aufleuchten. Mit dem Befehl HOME wird der Bildschirm vollständig gelöscht. Der Befehl TEXT:HOME schaltet den Computer auf die Textform um und löscht sofort den Bildschirm. Mit GR:HOME wird die Grafik auf dem Bildschirm gelöscht, aber der Rechner verbleibt in der LO-RES-Grafikform.

Probieren Sie, einige Parameter in dem FARBPUNKTE-Programm zu ändern. Was geschieht?

Frage 1: Wie muß FARBPUNKTE verändert werden, damit alle Punkte mit der Farbe rosa gezeichnet werden?

Frage 2: Mit dem Programm FARBPUNKTE sollen gelbe Punkte in der oberen Hälfte des Bildschirms gezeichnet werden? Welche Änderungen sind notwendig?

Frage 3: Ein Quadrat mit zehn Punkten Seitenlänge im Zentrum des Bildschirms soll mit violetten Punkten ausgefüllt werden. Welche Befehle sind notwendig?

Sicherlich können einige dieser Variationen des FARBPUNKTE-Programms in eigenen Programmen des Lesers angewandt werden. Wie können wir z.B. die Augen eines wilden Tieres darstellen, die nachts im dunklen Dschungel aufleuchten? Wir nehmen ein dunkles Grün und gelbe Farbpunkte und lassen diese auf dem dunklen Bildschirm erscheinen. Ein weiteres Beispiel: Wir schauen von oben auf Münzen, die in einen Wunschbrunnen geworfen werden. Dies wird dargestellt durch gelbe Farbpunkte, die in einem kreisförmigen Bereich im Zentrum des Bildschirms erscheinen. In beiden Fällen kann die Anzahl der Farbpunkte, die auf dem Bildschirm erscheinen, durch eine FOR-NEXT-Schleife kontrolliert werden.

Hier ist das vollständige Listing von WUNSCHBRUNNEN:

```
10 REM ...WUNSCHBRUNNEN...
11 :
100 GR : HOME
120 RA = 16: REM RADIUS DES BRUNNENS
130 R2 = RA * RA: REM QUADRAT DES RADIUS
140 H0 = 19: REM H-POS DES MITTELPUNKTS
150 V0 = 19: REM V-POS
200 PRINT
```

```

210 PRINT "DRUECKE EINE BELIEBIGE TASTE ZUM          BEENDEN DES PROGRAMMS
..";
300 REM  BESTIMME ERST H,V IN IN EINEM QUADRAT MIT ZENTRUM IN (H0,V0) U
ND SEITENLAENGE = 2*RA
310 H = INT (2 * RA * RND (1)) + H0 - RA
400 V = INT (2 * RA * RND (1)) + V0 - RA
410 REM  DANN KONTOLLIERE, OB (H,V) IN DER MITTE DES KREISES IST
420 IF (H - H0) ^ 2 + (V - V0) ^ 2 > R2 THEN 310: REM  BESTIMME EINEN N
EUEEN PUNKT, WENN DER ERSTE IM QUADRAT,ABER NICHT IM KREIS IST
500 COLOR= 13
600 PLOT H,V
700 P = 1
710 FOR Z = 1 TO P: NEXT : REM  PAUSE
800 IF PEEK ( - 16384) < 128 THEN 300: REM  KEINE TASTE GEDRUECKT
810 GET Z$: REM  LOESCHE  TASTENDRUCK

```

Dies ist nur eine kleine Auswahl der Möglichkeiten, die mit einem Programm zur Erzeugung von Farbpunkten programmiert werden können.

Ein weiterer Weg zur Änderung des Programms ergibt sich durch Verwendung einer variablen Zeitverzögerung; sie wird durch den Wert der Variablen P in Zeile 700 angegeben. Dazu wählen wir einen Intervallbereich, in dem der Parameter P enthalten sein soll. Nun benutzen wir die Anweisung, die auf Seite 18 gegeben wurde.

Frage 4: Es sollen nur Punkte in den Farben blau und orange gezeichnet werden. Wie muß das Programm geändert werden? (Siehe Farbtabelle auf Seite 17)

Frage 5: Es sollen abermals Punkte in den Farben blau und orange gezeichnet werden. Dabei soll die Farbe blau dreimal so häufig erscheinen wie die Farbe orange. Welche Befehle sind notwendig?

Das folgende Programm simuliert das "Wachsen" von Blumen auf einem leeren Feld. Für die Blumen haben wir drei Farben gewählt (rot, gelb und violett). Jede dieser Farben besitzt eine Wahrscheinlichkeit von 20 %. Die Farbe grün hat eine Wahrscheinlichkeit von 40 %.

Verändern Sie das FARBPUNKTE-Programm wie folgt:

```

10 REM  ...BLUMEN - FARBPUNKTE...
11 :
400 V = INT (30 * RND (1)) + 10
500 Z = INT (100 * RND (1)) + 1: REM  1/100
510 COLOR= (Z < = 20) * 1 + (Z > 20 AND Z < = 40) * 13 + (Z > 40 AND Z
< = 60) * 3 + (Z > 60) * 4

```

Speichern Sie dieses Programm als BLUMEN.

Beobachten Sie, wie die Blumen den Bildschirm bedecken. Um sicherzustellen, daß die Blumen nicht den Himmel bedecken, haben wir die V-Koordinate begrenzt. Dadurch erscheinen keine Punkte oberhalb der Koordinate V = 10 (siehe auch Zeile 400 in obigem Listing).

Farbige Linien

Der Unterschied zwischen dem Zeichnen von Punkten und dem Zeichnen von Linien ist folgender: für Punkte werden zwei Koordinaten benötigt, für Linien müssen beide Endpunkte angegeben werden. Da wir nur horizontale oder vertikale Linien zeichnen lassen, ist die Angabe der Endpunkte einfach.

Bei dem Befehl HLIN werden die Spalten angegeben, wo die Linie beginnt bzw. endet. Weiter muß die vertikale Entfernung vom oberen Rand des Bildschirms festgesetzt werden. HLIN 10,20 AT 5 ergibt eine horizontale Linie in Zeile 5, die von Spalte 10 bis Spalte 20 verläuft. VLIN 10,20 AT 5 ergibt eine vertikale Linie in Spalte 5; Sie reicht von Zeile 10 bis zur Zeile 20.

Im folgenden Programm werden Linien statt Punkte gezeichnet. Es baut auf den bisher gezeigten Programmen auf. Das Programm wählt die Endpunkte selbst aus und bestimmt, ob die Linie horizontal oder vertikal verläuft. Außerdem wird die Farbe der Linien und die Zeitverzögerung zwischen dem Zeichnen zweier Linien gewählt.

Ändern Sie das FARBPUNKTE-Programm wie folgt ab:

```
10 REM    ...FARBIGE LINIEN-FARBPUNKTE
11 :
300 H1 = INT (40 * RND (1))
350 H2 = INT (40 * RND (1))
400 V1 = INT (40 * RND (1))
450 V2 = INT (40 * RND (1))
600 D = INT (2 * RND (1)): REM D =0 (HLIN); =1 (VLIN)
650 IF D = 0 THEN HLIN H1,H2 AT V1
660 IF D = 1 THEN VLIN V1,V2 AT H1
```

Speichern Sie das Programm als FARBIGE LINIEN und starten Sie es. Variieren Sie einige Parameter. Wie wird die Anzahl der Farben begrenzt (Zeile 500)?

Die folgenden Änderungen des Programms FARBIGE LINIEN ermöglichen es, die Länge der Linien zu begrenzen.

```
10 REM    ...LAENGE LINIEN-FARBIGE LINIEN...
11 :
120 HL = 20: REM    MAXIMUM HLIN LAENGE
130 VL = 10: REM    VLIN
360 IF ABS (H2 - H1) > HL THEN 350: REM    NOCH EINMAL - ZU LANG
460 IF ABS (V2 - V1) > VL THEN 450: REM    NOCH EINMAL - ZU LANG
```

Speichern Sie dieses Programm als LAENGE LINIEN.

Das Programm kann so variiert werden, daß der Bildschirm mit kurzen vertikalen Linien ausgefüllt wird. Dazu müssen lediglich die horizontalen Linien eliminiert werden, was durch Änderung von Zeile 600 geschieht:

600 D = 1.

Noch interessanter wird das Programm, wenn wir es interaktiv gestalten. Bis jetzt wurden alle Variablen vom Programm selbst bestimmt. Es ist leicht dahingehend zu verändern, daß die Variablenwerte über die Tastatur eingelesen werden. Machen Sie folgende Änderungen am Programm LAENGE LINIEN:

```

10 REM   ...EINGABE FARBE - LAENGE LINIEN...
11 :
150 CL = 1: REM   ANFANGSFARBE
210 PRINT "DRUECKE ESC FUER PROGRAMMENDE ...";
500 Z$ = "ASDFGHJKLZXCVBNM"
510 GOSUB 1000: REM   UEBERPRUEFE TASTENDRUCK
520 IF Z THEN CL = Z - 1: REM   NEUE FARBE BEI TASTENDRUCK
530 COLOR= CL
800 Z$ = CHR$ (27): REM   UEBERPRUEFE, OB ESC GEDRUECKT
810 GOSUB 1000
820 IF Z = 0 THEN 300
900 END
991 :
992 REM   * UEBERPRUEFE, OB TASTEN VORGESEHEN
993 REM   ENTRY: Z$ TEXT, GIBT VERWENDETE TASTEN AN
994 REM   EXIT: Z   0 (KEINE ZUORDNUNG) UND TASTENDRUCK (WENN VORHANDEN)
      NICHT GELOESCHT
995 REM           Z   J (J'TER ZEICHEN IN Z$) UND TASTENDRUCK GELOESCHT
999 :
1000 Z = 0: REM   SETZE FLAG KEINE ZUORDNUNG
1010 Z1 = PEEK ( - 16384) - 128: REM   LESE TASTENDRUCK
1020 IF Z1 < 0 THEN RETURN : REM   KEINE TASTE GEDRUECKT
1030 IF LEN (Z$) = 0 THEN RETURN : REM   KEINE ZEICHEN VORHANDEN
1040 FOR Z2 = 1 TO LEN (Z$)
1050 IF Z1 = ASC ( MID$ (Z$,Z2,1)) THEN Z = Z2: GET Z1$: REM   ZUORDNUN
      G ERLEDIGT-TASTENDRUCK GELOESCHT
1060 NEXT
1070 RETURN

```

Speichern Sie dieses Programm als EINGABE FARBEN. Auf dem Bildschirm erscheinen nur rote Linien. Drücken Sie, während das Programm abläuft, einige der Tasten in den beiden unteren Reihen. (A bis L oder Z bis M.) Jedes Betätigen einer Taste ändert die Farbe der ausgedruckten Linien. Wir haben jeder Taste eine der APPLE-Farben zugeordnet (Zeile 500). Die Zuordnung ist natürlich völlig beliebig. Als Zuordnungsschema haben wir die Reihenfolge der Farben im Regenbogen benutzt. Ebenso gut kann jedes andere Schema benutzt werden.

Das Programm erhält die Variablenwerte über die Tastatur. Die Zahlenwerte können während des Programmablaufs direkt geändert werden. Mit diesem Prinzip können weitere Änderungen vorgenommen werden. Wir bestimmen die Länge der horizontalen Linien über die Tasten 1 bis 9, und die vertikalen Linien mit Hilfe der Tasten Q bis O. Die Tasten auf der linken Seite der Tastatur produzieren kurze Linien. Die Tasten auf der rechten Seite ergeben längere Linien.

Ändern Sie EINGABE FARBEN wie folgt:

```
10 REM ...EINGABE LAENGEN-EINGABE FARBEN...
11 :
310 Z$ = "123456789"
320 GOSUB 1000
330 IF Z THEN HL = 2 * Z: REM NEUE HLIN-LAENGE, WENN TASTE GEDRUECKT
340 Z = 1 - 2 * INT (2 * RND (1)): REM +1, -1
350 H2 = H1 + Z * HL: REM + OR - HL
360 IF H2 < 0 OR H2 > 39 THEN H2 = H1 - Z * HL: REM - ODER + , WENN AUS
    SERHALB DES VORGESEHEN INTERVALLS
410 Z$ = "QWERTYUIO"
420 GOSUB 1000
430 IF Z THEN VL = 2 * Z: REM NEUE VLIN-LAENGE, WENN TASTE GEDRUECKT
440 Z = 1 - 2 * INT (2 * RND (1)): REM +1, -1
450 V2 = V1 + Z * VL: REM + OR - VL
460 IF V2 < 0 OR V2 > 39 THEN V2 = V1 - Z * VL: REM - ODER + , WENN AUS
    SERHALB DES GUELFIGEN BEREICHS
```

Speichern Sie dieses Programm unter dem Namen EINGABE LAENGE. Es macht sehr viel Freude, mit dem Programm interaktiv zu arbeiten, denn so besteht eine unmittelbare Kontrolle über das, was der Computer zeichnet und auf dem Bildschirm demonstriert. Wichtig beim Entwerfen von Programmen ist deshalb die Interaktivität, denn das Spielen mit interaktiven Programmen bereitet dem Benutzer mehr Spaß.

Die Farbpunkte, die bei der LO-RES-Grafik auf dem Bildschirm erscheinen, sind nicht exakt quadratisch. Die waagerechte Ausdehnung ist größer als die senkrechte. Dies folgt aus der Struktur der LO-RES-Hardware. Aus dem gleichen Grunde sind vertikale Linien etwas dicker und kürzer als horizontale Linien, obwohl die Längensvariablen bei beiden den gleichen Wert haben.

Wie leicht zu erkennen ist, können in diesem Programm durch Variation von Parametern viele weitere interessante Effekte erzielt werden: Die Anzahl der Farben kann begrenzt werden, oder die horizontalen und vertikalen Längen können auf andere Art und Weise den Tasten zugeordnet werden.

Der nächste Schritt ist das Zeichnen von Rechtecken. Dabei zeichnen wir zuerst eine Linie und geben an, wie weit und in welche Richtung die Linie verbreitert werden soll. Das letzte Programm dieses Abschnitts zeichnet farbige Rechtecke. Ändern Sie EINGABE LAENGE wie folgt:

```
10 REM ...EINGABE RECHTECKE-EINGABE LAENGE
11 :
650 IF D = 0 THEN FOR Z = V1 TO V2 STEP SGN (V2 - V1): HLIN H1,H2 AT Z
    : NEXT
660 IF D = 1 THEN FOR Z = H1 TO H2 STEP SGN (H2 - H1): VLIN V1,V2 AT Z
    : NEXT
```

Speichern Sie dieses Programm als EINGABE RECHTECKE.

Mit den Tasten 1 bis 9 kann die Breite der Rechtecke variiert werden, die Tasten Q bis O ändern die Höhe und die Tasten A bis M die Farben. Wir sehen daher einen direkten Zusammenhang zwischen dem Betätigen der Tasten und der Bildschirmwiedergabe.

Weitere Farbprogramme

Hier stellen wir weitere Möglichkeiten vor, den Bildschirm farbig zu gestalten. Zuerst zeigen wir ein Programm zum Zeichnen eines farbigen Randes. Dies ist nützlich, um die Aufmerksamkeit auf die Bildschirmwiedergabe zu lenken. Das Programm erzeugt einen farbigen Rand von der Breite einer Zeile um den Bildschirm herum. Die Farbe des Randes wird in Zeile 500 festgelegt.

Speichern Sie dieses Programm als RAND1.

Wir können RAND1 dahingehend ändern, daß es Ränder verschiedener Farbe zeichnet, die aufeinander folgen. Hier sind beide Programme.

```
10 REM ...BORDER...
11 :
100 GR : HOME
500 COLOR= 1
600 HLIN 0,39 AT 0
610 VLIN 1,39 AT 39
620 HLIN 38,0 AT 39
630 VLIN 38,1 AT 0
```

Durch Löschen der Zeilen 600 bis 630 und Hinzufügen der folgenden Zeilen erhalten wir ein allgemeiner verwendbares Programm.

```
10 REM ...RAND2 - RAND1...
11 :
600 Z = 0: REM 0 PUNKTE VOM RAND AUS
610 GOSUB 900
820 END
891 :
892 REM * LOW-RES RAND *
893 REM EINGANG: Z ANZAHL DER PUNKTE VOM RAND AUS
894 REM FARBE GESETZT
899 :
900 Z1 = 39 - Z
910 HLIN Z,Z1 AT Z
920 VLIN Z + 1,Z1 AT Z1
930 HLIN Z1 - 1,Z AT Z1
940 VLIN Z1 - 1,Z + 1 AT Z
950 RETURN
```

Speichern Sie dieses Programm als RAND2.

Frage 6: Wie muß RAND2 geändert werden, damit die Umrandung des Bildschirms drei Zeilen nach innen eingerückt wird?

Frage 7: Wie lauten die Befehle zum Zeichnen einer doppelten Umrandung, wobei sich zwischen den beiden Rändern noch freier Raum befindet?

Ein Programm, das in der Lage ist, den Bildschirm zu säubern (d.h. schnell und fließend mit Farbe auszufüllen), kann sehr nützlich sein. Das folgende Programm erreicht dies durch das Zeichnen horizontaler Linien.

```

10 REM ...WASCHEN...
11 :
100 GR : HOME
500 COLOR= 1
600 FOR Z = 0 TO 39
610 HLIN 0,39 AT Z
620 NEXT

```

Speichern Sie dieses Programm als WASCHEN.

Um Streifen in wechselnder Farbe zeichnen zu lassen, wird das Programm WASCHEN wie folgt geändert:

```

10 REM ...STREIFEN - WASCHEN
11 :
500 C1 = 3: REM ERSTE FARBE
510 C2 = 7: REM ZWEITE FARBE
520 C = C1: REM AKTUELLER PARAMETER
605 COLOR= C:C = C1 + C2 - C: REM GEHE ZU ANDERER FARBE UEBER

```

Das WASCHEN-Programm stellt eine Hintergrundfarbe zur Verfügung. Über diesen Hintergrund können wir andere Linien, Punkte und sogar Bilder zeichnen, was wir im nächsten Kapitel zeigen. Die Hintergrundfarbe im WASCHEN-Programm kann durch Änderung der Zeile 500 gewechselt werden.

Ein weiterer Weg, den Bildschirm farbig auszufüllen, ist das Zeichnen von Streifen in Spiralenform:

```

10 REM ...SPIRALE...
11 :
100 GR : HOME
120 H0 = 19: REM H-POS DES ZENTRUMS
130 V0 = 19: REM V-POS
140 N = 19: REM ZAHL DER SPIRALWINDUNGEN
150 P = 1: REM PAUSE ZWISCHEN DEM ZEICHNEN DER SEGMENTE
200 FOR J = N TO 0 STEP - 1
210 H1 = H0 - J: REM LINKER RAND DES AKTUELLEN SPIRALARMS
220 H2 = H0 + J + 1: REM RECHTER RAND
230 V1 = V0 - J: REM OBERER RAND
240 V2 = V0 + J + 1: REM UNTERER RAND
300 GOSUB 900
310 HLIN H1,H2 AT V1
350 GOSUB 900
360 VLIN V1 + 1,V2 AT H2
400 GOSUB 900
410 HLIN H2 - 1,H1 AT V2
450 GOSUB 900
460 VLIN V2 - 1,V1 + 1 AT H1
490 NEXT
820 END
892 :
893 REM * WAEHLE FARBE FUER NAECHSTES SEGMENT,DANN ZEITVERZOEGERUNG*
899 :
900 COLOR= 1
910 FOR Z = 1 TO P: NEXT
990 RETURN

```

Speichern Sie dieses Programm als SPIRALE.

Wir können die Zeichengeschwindigkeit verlangsamen, um das Entstehen der Spirale besser beobachten zu können. Die Zeitverzögerung wird in Zeile 150 verändert:

150 P = 200

Frage 8: Wie muß SPIRALE geändert werden, damit jeder Spiralarm in einer anderen Farbe gezeichnet wird?

Speichern Sie das Programm mit dieser Änderung als SPIRALE1.

Hier ist eine weitere Variation von SPIRALE1. Sie bewirkt, daß die Spirale sich schließt und anschließend wieder öffnet:

```
10 REM ...SPIRALE2-SPIRALE1...
11 :
190 PRINT : PRINT "DRUECKE ESC ZUM BEENDEN DER GRAFIK..";
500 COLOR= 0: FOR J = 0 TO N
510 H1 = H0 - J
520 H2 = H0 + J + 1
530 V1 = V0 - J
540 V2 = V0 + J + 1
600 GOSUB 910
610 VLIN V1 + 1,V2 - 1 AT H1
650 GOSUB 910
660 HLIN H1,H2 - 1 AT V2
700 GOSUB 910
710 VLIN V2,V1 + 1 AT H2
750 GOSUB 910
760 HLIN H2,H1' AT V1
790 NEXT
800 IF PEEK ( - 16384) < 128 THEN 200
810 GET Z$: IF Z$ < > CHR$ (27) THEN 200: REM ESC NICHT GEDRUECKT
```

Speichern Sie das Programm als SPIRALE2. Schauen Sie, was das Programm auf dem Bildschirm bewirkt.

Mit folgender Änderung von SPIRALE2 wird die Größe der Spirale reduziert. Außerdem kann der Mittelpunkt der Spirale zu anderen Stellen des Bildschirms bewegt werden.

```
10 REM ...SPIRALE3-SPIRALE2...
11 :
120 H0 = 10
130 V0 = 12
140 N = 4
```

Speichern Sie dieses Programm als SPIRALE3.

Wir wollen nun zwei Spiralen darstellen, die sich gleichzeitig öffnen und schließen. Dazu muß das folgende Programm zu SPIRALE3 hinzugefügt werden. (Dies ist zwar eine langwierige Arbeit, ergibt aber einen sehr wirkungsvollen Effekt.)

```

10 REM    ..ZWEI SPIRALEN-SPIRALE3...
11 :
122 HC = 29: REM    ABSTAND ZU SPIRALE 2
132 VC = V0
212 HA = HC - J
222 HB = HC + J + 1
232 VA = VC - J
242 VB = VC + J + 1
312 HLIN HA,HB AT VA
362 VLIN VA + 1,VB AT HB
412 HLIN HB - 1,HA AT VB
462 VLIN VB - 1,VA + 1 AT HA
512 HA = HC - J
522 HB = HC + J + 1
532 VA = VC - J
542 VB = VC + J + 1
612 VLIN VA + 1,VB - 1 AT HA
662 HLIN HA,HB - 1 AT VB
712 VLIN VB,VA + 1 AT HB
762 HLIN HB,HA AT VA

```

Speichern Sie das Programm unter dem Namen ZWEI SPIRALEN.

Mit etwas Vorstellungskraft können diese Effekte als zwei Augen gesehen werden. Vielleicht benötigt einer unserer Leser ein dreiäugiges Monster zum Verschönern seines Programms!

Kombination von Klang und Farbe

Wir wollen nun eines der Farbprogramme zusätzlich mit Klangeffekten ausstatten. Das folgende Programm spielt eine auf- und absteigende Folge von Tönen und zeigt die Spirale, die sich entsprechend öffnet und schließt.

Ändern Sie SPIRALE2 wie folgt:

```

10 REM    ...SPIRALE KLANG-SPIRALE2...
11 :
180 WD = 10
920 WP = N + 1 - J
930 GOSUB 13000

```

Verbinden Sie das Programm mit KLANG EINHEIT und speichern es unter dem Namen SPIRALE KLANG.

Durch eine zusätzliche Änderung erhalten wir ein Programm, das den Spieler völlig verblüfft. Die Töne sind dem Rhythmus der Spirale nicht mehr ganz angepaßt, d.h. beide beginnen und enden nicht mehr gleichzeitig.

```

10 REM    ...LUSTIGE SPIRALE - SPIRALE KLANG...
11 :
170 CP = 1:CZ = .25
920 WP = INT (CP)
940 CP = CP + CZ
950 IF CP > N + 2.25 THEN CZ = - .25: GOTO 940: REM    ABWAERTS
960 IF CP < 1 THEN CZ = .25: GOTO 940: REM    AUFWAERTS

```

Es bieten sich eine Reihe weiterer Experimente an. Wir können LO-RES-Grafikeffekte zum ORGEL-Programm hinzufügen. Zum Beispiel: Jedesmal wenn wir eine Note drücken, erscheint ein farbiger Punkt. Oder wir programmieren spezielle Grafikeffekte, falls eine der Ziffern gedrückt wird. Das Drücken der Taste 1 könnte ein Bedecken des Bildschirms mit roter Farbe hervorrufen, bei Taste 2 ist die Farbe orange, usw. Wird eine weitere zufällige Taste gedrückt, so könnte eine Spirale erscheinen.

Andererseits sind auch einfache Effekte denkbar, wie das bloße Wechseln einer zufällig bestimmten Farbe bei Betätigen der Taste 1.

Wir sehen also, es gibt eine große Anzahl von Möglichkeiten, Klänge und Farbe zu kombinieren. Auf diese Weise wird der Einsatz von beiden wirkungsvoller und interessanter.

LO-RES-Grafik auf dem ganzen Bildschirm

Jedes unserer bisherigen Programme verwendete ein vier Zeilen großes Textfenster am unteren Ende des Bildschirms. Mit den folgenden beiden Programmzeilen wird das Textfenster gelöscht. Wir erhalten acht zusätzliche Grafikzeilen.

```
10 REM    ..VOLLE LO-RES GRAFIK
11 :
100 GR : POKE - 16302,0: REM  SETZE LO-RES GRAFIK(GANZER BILDSCHIRM)
110 COLOR= 0: FOR Z = 40 TO 47: HLIN 0,39 AT Z: NEXT : REM  LOESCHE DIE
    8 UNTERSTEN ZEILEN
```

Zusammenfassung von Kapitel 2

Dieses Kapitel stellte die LO-RES-Grafikbefehle vor und demonstrierte einige einfache Anwendungen. Die Programme WASCHEN, RAND und SPIRALE sind für eigene Programme des Lesers besonders nützlich.

Lösungen zu Kapitel 2

Frage 1: 500 COLOR = 11

Frage 2: Folgende Zeilen sind zu dem FARBPUNKTE-Programm hinzuzufügen:

```
10 REM ...GELBE STERNE-FARBPUNKTE...
11 :
400 V = INT (20 * RND (1))
500 COLOR= 13
700 P = 1000
```

Frage 3: Die Farbe wird zu violett bestimmt. Die H- und V-Koordinate werden so bestimmt, daß sie zwischen 15 und 24 variieren.

```
10 REM ...VIOLETTES RECHTECK-FARBPUNKTE...
11 :
300 H = INT (10 * RND (1)) + 15
400 V = INT (10 * RND (1)) + 15
500 COLOR= 3
700 P = 1000
```

Frage 4: 500 COLOR = 6
510 IF RND(1) .5 THEN CLOLOR = 9
(Die Zahl .5 gibt jeder Farbe eine Chance von 50 %.)

Frage 5: 500 COLOR = 6
510 IF RND(1) .25 THEN COLOR = 9
(Durch die Zahl .25 erhält die Farbe orange nur eine Chance von 25 %.)

Frage 6: 600 Z = 3

Frage 7: 620 Z = 2
630 GOSUB 900

Frage 8: 900 COLOR = INT(16*RND(1))

Kapitel 3

Bilder in LO-RES-Grafik

Dieses Kapitel behandelt speziell das Erstellen von Bildern und Symbolen in LO-RES-Grafik. Die geschaffenen Bilder können gespeichert und in späteren Programmen wiederverwendet werden. Wir werden einige Bilder vorstellen und zeigen, wie sie zu gebrauchen sind. Anschließend demonstrieren wir, wie Bilder hergestellt, verändert und gespeichert werden. Zu guter Letzt zeigen wir Programme, die verschiedene Bilder miteinander verbinden und verändern können. Der Grundbaustein eines Bildes ist der im letzten Kapitel eingeführte Farbpunkt. Alle Bilder in LO-RES-Grafik sind aus diesen viereckigen Farbpunkten zusammengesetzt. Ihre Qualität entspricht deshalb eher der von Kinderzeichnungen oder der von Bildern, die mit Kreuzstichen gestickt wurden. Besonders Kinder finden Bilder in LO-RES-Grafik sehr reizvoll.

BILD EINHEIT

Das folgende Programm BILD EINHEIT gibt uns die Möglichkeit, Bilder auf dem Fernsehschirm darzustellen. Wir bestimmen lediglich ein Bild, seine Position und Farbe; das Programm erledigt die ganze Arbeit für uns. Um die Benutzung des Programms anfangs etwas zu erleichtern, haben wir die Buchstaben des Alphabets und die Zahlen 1 bis 9 bereits in das Programm aufgenommen. Später werden wir zeigen, wie zusätzliche Bilder hergestellt, gespeichert und auf dem Monitor dargestellt werden. Wie in dem folgenden Listing leicht zu erkennen ist, hat das Setzen von Leerstellen entscheidenden Einfluß auf das Aussehen der Buchstaben und Zahlen. Beim Eintippen des Programms ist deshalb besonders für den ‚Bildteil‘ große Aufmerksamkeit erforderlich, damit die Wiedergabe der Zeichen richtig erfolgt.

```

10 REM ...BILD EINHEIT-NEXTDATA EINHEIT..
11 :
12 REM BILD UNTERPROGRAMME + BILD BIBLIOTHEK
13 :
14981 :
14982 :
14983 REM * DRUCKE BILD IN LOW-RES *
14984 REM BEGINN: XH H-POS DER OBEREN LINKEN ECKE
14985 REM XV V-POS
14986 REM XA HORIZONTALE BREITE
14987 REM XB VERTIKALE HOEHE
14988 REM LESE DATA-ANWEISUNGEN FUER ZEICHEN
14989 REM XC() WAEHLT DIE FARBEN
14990 REM GRAFIKFORMAT WIRD GEWAEHLT
14991 REM AUSGANG: Z% 0 BILD IST PASSEND
14992 REM 1 FEHLER-BILD NICHT PASSEND
14999 :
15000 IF XH + XA > 40 OR XV + XB > 48 THEN Z% = 1: RETURN : REM FEHLER - BIL
D NICHT PASSEND
15010 Z1 = XV: REM ERSTE V-POS
15020 REM INITIALISIERUNG UNNOETIG- ERSTE ANGABE UEBER XC() BEWIRKT "DIM XC
(10)"
15030 READ Z$: IF Z$ = "-1" THEN Z% = 0: RETURN : REM UEBERPRUEFE OB DURCHG
EFUEHRT
15040 FOR Z = 1 TO LEN (Z$): REM DRUCKE JEDES DER 1-9 ZEICHEN
15050 Z% = ASC ( MID$ (Z$,Z,1)) - 48
15060 IF Z% > = 1 AND Z% < = 9 THEN COLOR= XC(Z%): PLOT XH + Z - 1,Z1: REM
DRUCKE FARBPUNKT
15070 NEXT
15080 Z1 = Z1 + 1: REM NAECHSTE V-POS
15090 GOTO 15030
15092 :
15093 REM * LESE ZEICHEN *
15094 REM EINGANG: Z ZEICHEN #
15095 REM AUSGANG: XA HORIZONTALE BREITE
15096 REM XB VERTIKALE HOEHE
15097 REM SETZE DATA-ZEIGER AUF BILDANFANG
15099 :
15100 Z = 20000 + 100 * Z: GOSUB 19000: REM SETZE DATA-ZEIGER
15110 READ XA,XB: REM DIE ERSTEN BEIDEN DATEN SIND BREITE UND HOEHE
15120 RETURN
15191 :
15192 REM * DRUCKE EIN BILD *
15193 REM EINGANG: Z BILD #
15194 REM XH H-POS VON ULHC
15195 REM XV V-POS
15196 REM FARBEN WERDEN MIT XC() GEWAEHLT
15197 REM AUSGANG: Z% 0 BILD IST PASSEND
15198 REM 1 FEHLER-BILD NICHT PASSEND
15199 :
15200 GOSUB 15100: REM SETZE DATA-ZEIGER
15210 GOTO 15000: REM DRUCKE BILD
15292 :
15293 REM * ZENTRIERE ZEICHENFOLGE *
15294 REM EINGANG: X$ TEXT
15295 REM XV V-POS VON ULHC
15296 REM FARBEN WERDEN IN XC() GEWAEHLT
15297 REM AUSGANG: Z% 0 BILD PASSEND
15298 REM 1 FEHLER-BILD NICHT PASSEND
15299 :
15300 IF LEN (X$) = 0 THEN RETURN : REM LEERER TEXT

```

```

15310 IF XS = 0 THEN XS = 1: REM INITIALISIERE RAUM ZWISCHEN DEN BILDERN
15320 X1 = - XS: REM INITIALISIERE LOW-RES BREITE
15330 FOR X = 1 TO LEN (X$)
15340 Z = ASC ( MID$ (X$,X,1)): GOSUB 15100: REM BILD #S IST IDENTISCH MIT
      ASCII #S
15350 X1 = X1 + XA + XS: REM NEUE LOW-RES BREITE
15360 NEXT
15370 IF X1 > 40 + XS THEN Z% = 1: RETURN : REM FEHLER - BILD NICHT PASSEND
15380 XH = 19 - INT (X1 / 2): REM DRUCKE AM LINKEN BILDRAND
15390 :
15391 REM * DRUCKE ZEICHENFOLGE *
15392 REM EINGANG: X$ TEXT
15393 REM XH H-POS VON ULHC
15394 REM XV V-POS
15395 REM FARBEN WERDEN MIT XC() GEWAHLT
15396 REM AUSGANG: XH NEU BERECHNET
15397 REM Z% 0 BILD PASSEND
15398 REM 1 FEHLER-BILD NICHT PASSEND
15399 :
15400 IF LEN (X$) = 0 THEN RETURN : REM LEERER TEXT
15410 IF XS = 0 THEN XS = 1: REM INITIALISIERE RAUM ZWISCHEN DEN BILDERN
15420 FOR X = 1 TO LEN (X$)
15430 Z = ASC ( MID$ (X$,X,1)): GOSUB 15200: REM DRUCKE EIN BILD AUS
15440 XH = XH + XA + XS: REM BERECHNE NEUE H-POS
15450 NEXT
15460 RETURN
15492 :
15493 REM * LOESCHE 40X40 BILDSCHIRM MIT EINER FARBE *
15494 REM EINGANG: FARBENSATZ
15499 :
15500 Z = 39: REM HOEHE
15510 FOR Z1 = 0 TO Z: HLIN 0,39 AT Z1: NEXT
15520 RETURN
24800 DATA 5,7: REM 0
24810 DATA " 111"
24820 DATA "1 1"
24830 DATA "1 11"
24840 DATA "1 1 1"
24850 DATA "11 1"
24860 DATA "1 1"
24870 DATA " 111"
24880 DATA "-1"
24900 DATA 5,7: REM 1
24910 DATA " 1"
24920 DATA " 11"
24930 DATA " 1"
24940 DATA " 1"
24950 DATA " 1"
24960 DATA " 1"
24970 DATA " 111"
24980 DATA "-1"
25000 DATA 5,7: REM 2
25010 DATA " 111"
25020 DATA "1 1"
25030 DATA " 1"
25040 DATA " 11"
25050 DATA " 1"
25060 DATA "1"
25070 DATA "11111"
25080 DATA "-1"
25100 DATA 5,7: REM 3

```

25110	DATA	"11111"	
25120	DATA	" 1"	
25130	DATA	" 1"	
25140	DATA	" 11"	
25150	DATA	" 1"	
25160	DATA	"1 1"	
25170	DATA	" 111"	
25180	DATA	"-1"	
25200	DATA	5,7: REM	4
25210	DATA	" 1"	
25220	DATA	" 11"	
25230	DATA	" 1 1"	
25240	DATA	"1 1"	
25250	DATA	"11111"	
25260	DATA	" 1"	
25270	DATA	" 1"	
25280	DATA	"-1"	
25300	DATA	5,7: REM	5
25310	DATA	"11111"	
25320	DATA	"1"	
25330	DATA	"1111"	
25340	DATA	" 1"	
25350	DATA	" 1"	
25360	DATA	"1 1"	
25370	DATA	" 111"	
25380	DATA	"-1"	
25400	DATA	5,7: REM	6
25410	DATA	" 111"	
25420	DATA	" 1"	
25430	DATA	"1"	
25440	DATA	"1111"	
25450	DATA	"1 1"	
25460	DATA	"1 1"	
25470	DATA	" 111"	
25480	DATA	"-1"	
25500	DATA	5,7: REM	7
25510	DATA	"11111"	
25520	DATA	" 1"	
25530	DATA	" 1"	
25540	DATA	" 1"	
25550	DATA	" 1"	
25560	DATA	"1"	
25570	DATA	"1"	
25580	DATA	"-1"	
25600	DATA	5,7: REM	8
25610	DATA	" 111"	
25620	DATA	"1 1"	
25630	DATA	"1 1"	
25640	DATA	" 111"	
25650	DATA	"1 1"	
25660	DATA	"1 1"	
25670	DATA	" 111"	
25680	DATA	"-1"	
25700	DATA	5,7: REM	9
25710	DATA	" 111"	
25720	DATA	"1 1"	
25730	DATA	"1 1"	
25740	DATA	" 111"	
25750	DATA	" 1"	
25760	DATA	" 1"	
25770	DATA	"111"	
25780	DATA	"-1"	
26500	DATA	5,7: REM	A
26510	DATA	" 1"	
26520	DATA	" 1 1"	
26530	DATA	"1 1"	
26540	DATA	"1 1"	
26550	DATA	"11111"	
26560	DATA	"1 1"	
26570	DATA	"1 1"	
26580	DATA	"-1"	
26600	DATA	5,7: REM	B
26610	DATA	"1111"	
26620	DATA	"1 1"	
26630	DATA	"1 1"	
26640	DATA	"1111"	
26650	DATA	"1 1"	
26660	DATA	"1 1"	
26670	DATA	"1111"	
26680	DATA	"-1"	
26700	DATA	5,7: REM	C
26710	DATA	" 111"	
26720	DATA	"1 1"	
26730	DATA	"1"	
26740	DATA	"1"	
26750	DATA	"1"	
26760	DATA	"1 1"	
26770	DATA	" 111"	
26780	DATA	"-1"	
26800	DATA	5,7: REM	D
26810	DATA	"1111"	
26820	DATA	"1 1"	
26830	DATA	"1 1"	
26840	DATA	"1 1"	
26850	DATA	"1 1"	
26860	DATA	"1 1"	
26870	DATA	"1111"	
26880	DATA	"-1"	
26900	DATA	5,7: REM	E
26910	DATA	"11111"	
26920	DATA	"1"	
26930	DATA	"1"	
26940	DATA	"1111"	
26950	DATA	"1"	
26960	DATA	"1"	
26970	DATA	"11111"	
26980	DATA	"-1"	
27000	DATA	5,7: REM	F
27010	DATA	"11111"	
27020	DATA	"1"	
27030	DATA	"1"	
27040	DATA	"1111"	
27050	DATA	"1"	
27060	DATA	"1"	
27070	DATA	"1"	
27080	DATA	"-1"	
27100	DATA	5,7: REM	G
27110	DATA	" 1111"	
27120	DATA	"1"	
27130	DATA	"1"	
27140	DATA	"1"	
27150	DATA	"1 11"	

27160	DATA	"1 1"	
27170	DATA	" 1111"	
27180	DATA	"-1"	
27200	DATA	5,7: REM	H
27210	DATA	"1 1"	
27220	DATA	"1 1"	
27230	DATA	"1 1"	
27240	DATA	"11111"	
27250	DATA	"1 1"	
27260	DATA	"1 1"	
27270	DATA	"1 1"	
27280	DATA	"-1"	
27300	DATA	3,7: REM	I
27310	DATA	"111"	
27320	DATA	" 1"	
27330	DATA	" 1"	
27340	DATA	" 1"	
27350	DATA	" 1"	
27360	DATA	" 1"	
27370	DATA	"111"	
27380	DATA	"-1"	
27400	DATA	6,7: REM	J
27410	DATA	" 111"	
27420	DATA	" 1"	
27430	DATA	" 1"	
27440	DATA	" 1"	
27450	DATA	" 1"	
27460	DATA	"1 1"	
27470	DATA	" 111"	
27480	DATA	"-1"	
27500	DATA	5,7: REM	K
27510	DATA	"1 1"	
27520	DATA	"1 1"	
27530	DATA	"1 1"	
27540	DATA	"11"	
27550	DATA	"1 1"	
27560	DATA	"1 1"	
27570	DATA	"1 1"	
27580	DATA	"-1"	
27600	DATA	4,7: REM	L
27610	DATA	"1"	
27620	DATA	"1"	
27630	DATA	"1"	
27640	DATA	"1"	
27650	DATA	"1"	
27660	DATA	"1"	
27670	DATA	"1111"	
27680	DATA	"-1"	
27700	DATA	7,7: REM	M
27710	DATA	"1 1"	
27720	DATA	"11 11"	
27730	DATA	"1 1 1 1"	
27740	DATA	"1 1 1"	
27750	DATA	"1 1"	
27760	DATA	"1 1"	
27770	DATA	"1 1"	
27780	DATA	"-1"	
27800	DATA	5,7: REM	N
27810	DATA	"1 1"	
27820	DATA	"1 1"	
27830	DATA	"11 1"	
27840	DATA	"1 1 1"	
27850	DATA	"1 11"	
27860	DATA	"1 1"	
27870	DATA	"1 1"	
27880	DATA	"-1"	
27900	DATA	5,7: REM	O
27910	DATA	" 111"	
27920	DATA	"1 1"	
27930	DATA	"1 1"	
27940	DATA	"1 1"	
27950	DATA	"1 1"	
27960	DATA	"1 1"	
27970	DATA	" 111"	
27980	DATA	"-1"	
28000	DATA	5,7: REM	P
28010	DATA	"1111"	
28020	DATA	"1 1"	
28030	DATA	"1 1"	
28040	DATA	"1111"	
28050	DATA	"1"	
28060	DATA	"1"	
28070	DATA	"1"	
28080	DATA	"-1"	
28100	DATA	5,7: REM	Q
28110	DATA	" 111"	
28120	DATA	"1 1"	
28130	DATA	"1 1"	
28140	DATA	"1 1"	
28150	DATA	"1 1 1"	
28160	DATA	"1 1 1"	
28170	DATA	" 11 1"	
28180	DATA	"-1"	
28200	DATA	5,7: REM	R
28210	DATA	"1111"	
28220	DATA	"1 1"	
28230	DATA	"1 1"	
28240	DATA	"1111"	
28250	DATA	"1 1"	
28260	DATA	"1 1"	
28270	DATA	"1 1"	
28280	DATA	"-1"	
28300	DATA	5,7: REM	S
28310	DATA	" 111"	
28320	DATA	"1 1"	
28330	DATA	"1"	
28340	DATA	" 111"	
28350	DATA	" 1"	
28360	DATA	"1 1"	
28370	DATA	" 111"	
28380	DATA	"-1"	
28400	DATA	5,7: REM	T
28410	DATA	"11111"	
28420	DATA	" 1"	
28430	DATA	" 1"	
28440	DATA	" 1"	
28450	DATA	" 1"	
28460	DATA	" 1"	
28470	DATA	" 1"	
28480	DATA	"-1"	
28500	DATA	5,7: REM	U
28510	DATA	"1 1"	

28520	DATA	"1	1"	28820	DATA	"1	1"
28530	DATA	"1	1"	28830	DATA	" 1 1"	
28540	DATA	"1	1"	28840	DATA	" 1"	
28550	DATA	"1	1"	28850	DATA	" 1 1"	
28560	DATA	"1	1"	28860	DATA	"1 1"	
28570	DATA	" 111"		28870	DATA	"1 1"	
28580	DATA	"-1"		28880	DATA	"-1"	
28600	DATA	5,7: REM	V	28900	DATA	5,7: REM	Y
28610	DATA	"1	1"	28910	DATA	"1 1"	
28620	DATA	"1	1"	28920	DATA	"1 1"	
28630	DATA	"1	1"	28930	DATA	" 1 1"	
28640	DATA	"1	1"	28940	DATA	" 1"	
28650	DATA	"1	1"	28950	DATA	" 1"	
28660	DATA	" 1 1"		28960	DATA	" 1"	
28670	DATA	" 1"		28970	DATA	" 1"	
28680	DATA	"-1"		28980	DATA	"-1"	
28700	DATA	7,7: REM	W	29000	DATA	5,7: REM	Z
28710	DATA	"1	1"	29010	DATA	" 1"	
28720	DATA	"1	1"	29020	DATA	"11111"	
28730	DATA	"1	1"	29030	DATA	" 1"	
28740	DATA	"1 1 1"		29040	DATA	" 1"	
28750	DATA	"1 1 1 1"		29050	DATA	" 1"	
28760	DATA	"11 11"		29060	DATA	" 1"	
28770	DATA	"1 1"		29070	DATA	"1"	
28780	DATA	"-1"		29080	DATA	"11111"	
28800	DATA	5,7: REM	X	29090	DATA	"-1"	
28810	DATA	"1	1"				

Geben Sie dieses Programm ein und speichern Sie es als BILD EINHEIT.

Darstellen von Buchstaben und Zahlen

Für die Wiedergabe eines LO-RES-Bildes sind drei Schritte erforderlich:

1. Angabe eines bestimmten Bildes
2. Festlegung der Bildschirmposition
3. Wahl einer Farbe.

Mit dem Programm BILD EINHEIT ist es sehr einfach, Zahlen oder Buchstaben an verschiedenen Stellen des Bildschirms darzustellen. Wir geben einfach den Inhalt einer Textvariablen (X\$) an, bestimmen den Abstand zum oberen Bildschirmrand (XV) und entscheiden, ob der Text in der Zeilenmitte oder am Rand stehen soll. Natürlich wählen wir auch eine Farbe (XC(1)). Soll der Text nicht in der Zeilenmitte stehen, muß zusätzlich angegeben werden, wo der Text beginnt. Dazu wird die Variable XH bestimmt, die den Abstand vom linken Bildschirmrand angibt.

Im nächsten Abschnitt erläutern wir, wie die Buchstaben farbig dargestellt werden. In den DATA-Anweisungen unseres Programms taucht sehr oft die '1' auf. Wir haben alle Zahlen und Buchstaben so entworfen, daß sie nur in einer Farbe dargestellt werden können. Die Farbe haben wir aber in diesem Programm nicht festgelegt. Wenn wir nun eines dieser Zeichen benutzen, bestimmen wir die Farbe, indem wir eine der sechzehn APPLE-LO-RES-Farben den 'Einsen' in den DATA-Anweisungen zuordnen. Wenn wir zum Beispiel die Farbe hellgrün möchten, geben wir den Befehl XC(1) = 12 ein.

Damit wird die APPLE-Farbe Nr. 12 den 'Einsen' in den DATA-Anweisungen zugeordnet. Wollen wir die Farbe rosa haben, lautet der neue Befehl XC(1) = 11. Später werden wir zeigen, wie Bilder entworfen werden, die aus mehr als einer Farbe bestehen.

Wir zeigen nun einige Möglichkeiten, wie mit dem Programm BILD EINHEIT Wörter dargestellt werden.

Fügen Sie folgende Zeilen zu BILD EINHEIT hinzu und starten Sie das Programm:

```
100 GR:HOME
110 X$ = "KATZE"
120 XV = 10
130 XC(1) = 3
140 GOSUB 15300
999 END
```

Der Text wird in der Farbe grün gedruckt (Zeile 130), und das obere Ende der Buchstaben ist in Reihe 10 (Zeile 120). Das Wort "KATZE" steht in der Mitte des Bildschirms. Dies ist der Fall, wenn das Unterprogramm mit Zeile 15300 aufgerufen wird.

Fügen Sie die folgenden Zeilen hinzu und starten Sie das Programm erneut:

```
145 :
150 XH = 5
160 XV = 20
170 XC(1) = 8
180 GOSUB 15400.
```

Diese zusätzlichen Zeilen haben einige Variablen verändert. Da X\$ nicht geändert wurde, wiederholt sich die Ausgabe des Wortes "KATZE". Versuchen Sie das Programm so abzuändern, daß Ihr Name in mehreren Farben und an verschiedenen Stellen auf dem Bildschirm gedruckt wird. Was geschieht, wenn die Worte sich überlappen?

Es ist wichtig, daß alle Variablen zur Positionierung eines Textes auf dem Bildschirm spezifiziert sind. Erfolgt der Eintritt in das Unterprogramm in Zeile 15300, wird der Text in die Mitte der angegebenen Zeile gerückt. Rufen wir das Unterprogramm erst mit Zeile 15400 auf, wird der Text nicht automatisch in die Zeilenmitte plazierte. In diesem Fall müssen wir angeben, in welcher Spalte der Text beginnen soll (siehe Zeile 150).

Frage 1: Was wird auf dem Bildschirm dargestellt, wenn wir BILD EINHEIT mit dem folgenden Programm verbinden?

```
100 GR:HOME
110 X$ = "KATZE"
120 XV = 20
130 XC(1) = 3
140 GOSUB 15300
150 XC(1) = 13
160 GOSUB 15300
999 END
```


Frage 2: Was geschieht, wenn die Zeile

145 GR:HOME

zu diesem Programm hinzugefügt wird?

Als hübsche Zugabe enthält BILD EINHEIT ein Unterprogramm zum Löschen des Bildschirms mit einer Farbe nach Wahl.

Fügen Sie die folgenden Zeilen zum jetzigen Programm hinzu, und starten Sie es erneut:

102 COLOR = 5

104 GOSUB 15500

Hin und wieder ist es wünschenswert, die Variable X\$ vom Tastenfeld einzulesen anstatt sie im Programmlisting festzusetzen. Wir wollen zum Beispiel nach einem Namen fragen und diesen anschließend in großen Lettern darstellen. Die Buchstaben sind sehr groß; deshalb kann es sein, daß manche Namen nicht auf den Bildschirmbereich passen. Ein Unterprogramm von BILD EINHEIT überprüft die Länge der Namen. Auf diese Weise kann vermieden werden, daß ein Name am Ende abgeschnitten wird.

Beim Eintritt ins Unterprogramm in Zeile 15300 (Plazierung in der Bildschirmmitte), prüft das Unterprogramm die Länge des Textes. Dieser wird nur dann ausgedruckt, wenn er in den Bildschirmbereich hineinpaßt. Falls dies nicht zutrifft, wird der Text nicht ausgedruckt. Wenn wir aber erst in Zeile 15400 ins Unterprogramm eintreten, wird der Text am Ende abgeschnitten, wenn er zu lang ist.

Vor dem Verlassen des Unterprogramms wird eine Variable Z% bestimmt. Diese Variable zeigt an, ob der Text in den Bildschirmbereich hineinpaßt. Ist Z% gleich 0, dann ist der Text passend und wird ausgedruckt. Z% gleich 1 bedeutet, daß der Text zu lang ist und nicht auf dem Bildschirm erscheint (bei Eintritt in Zeile 15300). Bei Eintritt in Zeile 15400 werden nur diejenigen Buchstaben erscheinen, die innerhalb des Bildschirmbereichs liegen.

Das folgende Programm überprüft die Variable Z%. Fügen Sie folgende Zeilen zu BILD EINHEIT hinzu:

100 GR:HOME

110 PRINT "GEBEN SIE BITTE IHREN RUFNAMEN EIN!";

120 INPUT X\$

130 XV = 10: XC(1) = 3

140 GOSUB 15300

150 IF Z% = 0 GOTO 200

160 PRINT "DAS WAREN ZU VIELE BUCHSTABEN!"

170 PRINT "BENUTZE BITTE WENIGER BUCHSTABEN."

180 GOTO 110

200 ... Fortsetzung des Programms ...

Starten Sie dieses Programm.

Es bleibt zu bemerken, daß wir ein klein wenig auf das Erscheinen eines jeden Buchstabens warten müssen. Das Ausdrucken dieser Art von Buchstaben dauert länger als bei gewöhnlichen Textbuchstaben (wie z.B. im Programm-listing). Die Buchstaben und Zahlen im Textformat werden mittels der internen Logik des Rechners sehr schnell erstellt. Dagegen müssen die Zeichen, die wir hier vorgestellt haben, Schritt für Schritt in einem BASIC-Programm aufgebaut werden. Dies dauert natürlich etwas länger.

Vorschläge für Spiele mit Buchstaben

Wir wollen nun zwei Spiele vorschlagen, die sich besonders eignen, die Fähigkeiten von Anfängern zu schulen. Sie helfen, das Erkennen von Zahlen zu lernen, und machen Leseanfänger besser mit der Tastatur des Computers vertraut.

Im ersten Spiel tippt der Spieler einen Buchstaben. Das Programm druckt dieses Zeichen mit der LO-RES-Grafik auf dem Bildschirm aus. Ein Fortgeschrittener kann dazu die Namen der Buchstaben nennen, wenn diese auf dem Monitor erscheinen. Auf diese Weise betreut er während des Spiels den beginnenden Leser und beschleunigt den Lernerfolg.

Beim zweiten Spiel wird zuerst eine Zahl ausgedruckt. Der Spieler wird angewiesen, die entsprechende Taste zu drücken. Dabei sollten alle anderen Tasten vom Programm ignoriert werden, um unnötige Verwirrung zu vermeiden. Wird die richtige Taste gedrückt, gibt der Rechner einen Ton von sich, und die nächste Zahl erscheint.

Bemerkungen zur Zeilennummerierung

Die DATA-Anweisungen in BILD EINHEIT, die unsere Buchstaben und Zahlen enthalten, beginnen in Zeile 24800. Beim Entwerfen der Zeichen und Zuordnen der entsprechenden Zeilennummern benutzten wir folgende Regeln. Jedes Zeichen beginnt bei einer Zeilennummer, die ein Vielfaches von 100 ist. Hier steht eine DATA-Anweisung für die Länge und Höhe des Zeichens, sowie eine REM-Anweisung, die uns angibt, um welches Zeichen es sich handelt. Jedes Zeichen endet mit : DATA “— 1“.

Durch diese spezielle Form der Zeilennummerierung erhalten wir einen sehr einfachen Zugang zu den einzelnen Zeichen. Deshalb ist es wichtig, dies gut zu verstehen. Zuerst subtrahieren wir von der Zeilennummer, mit der ein Zeichen beginnt, die Zahl 20000. Das Ergebnis ist eine Zahl, die genau das Hundertfache des ASCII-Wertes des betreffenden Zeichens beträgt. Zum Beispiel: der Buchstabe A beginnt in Zeile 26500. Subtrahieren wir 20000 von dieser Zahl, erhalten wir 6500. Dies ist genau das Hundertfache des ASCII-Wertes für A, denn dieser beträgt 65. Der ASCII-Wert für B ist 66. Deshalb beginnen die DATA-Anweisungen für den Buchstaben B in Zeile 26600.

Mit unserer speziellen Zeilennummerierung können wir Zeichen spezifizieren, indem wir die durch den ASCII-Code zugehörige Kennzahl benutzen. Die Kennzahl des Buchstabens A ist zum Beispiel 65. Das bedeutet aber, daß wir

ebenso alle anderen Zeichen der Tastatur herstellen können. Später erfolgt der Zugriff auf diese Zeichen durch Verwendung ihrer ASCII-Kennzahl.

Ergänzend zu den Großbuchstaben, die wir im Programm zur Verfügung stellen, kann der Leser auch Kleinbuchstaben entwerfen. Wir schlagen vor, hier mit Bild Nr. 97 zu beginnen (Zeile 29700). Die Kennzahl jedes Kleinbuchstabens ist dann gleich der Kennzahl des zugehörigen Großbuchstabens plus 32. Dies entspricht somit dem Standard-ASCII-Code. Die Wahl, ob Klein- oder Großbuchstabe erscheinen soll, kann durch eine Textvariable belegt werden.

Fügen Sie das folgende Unterprogramm zu BILD EINHEIT hinzu. Dieses Unterprogramm wandelt die ASCII-Zahl des Großbuchstabens in die zugehörige Kennzahl des Kleinbuchstabens um.

```
15591 :  
15592 REM *UMWANDLUNG VON GROSS- IN KLEINBUCHSTABEN*  
15593 REM EINTRITT: Z$ GROSSBUCHSTABE  
15594 REM AUSTRITT: Z1$ KLEINBUCHSTABE  
15599 :  
15600 Z1$ = " "  
15610 IF LEN(Z$) = 0 THEN RETURN: REM KEIN ZEICHEN  
15620 FOR Z = 1 TO LEN(Z$)  
15630 Z1$ = Z1$ + CHR$(ASC(MID$(Z$, Z, 1)) + 32)  
15640 NEXT  
15650 RETURN
```

Nachdem wir die Kleinbuchstaben zu BILD EINHEIT hinzugefügt haben, können wir zum Beispiel X\$ = "Katze" wie folgt schreiben:

```
500 Z$ = "ATZE": GOSUB 15600  
510 X$ = "K" + Z1$
```

Unsere Form der Zeilennumerierung bietet Platz für 255 Zeichen (von Zeile 20100 bis Zeile 45599). Die Nummern 32 bis 127 bleiben reserviert für die Zeichen des ASCII-Codes. Wir haben also genügend Raum für weitere Zeichen, die wir selbst entwerfen können.

Anlegen einer Zeichen-Bibliothek

Vielleicht möchte der Leser noch viele andere Zeichen zur Verfügung haben. Zu diesem Zweck empfehlen wir, mit dem Erstellen einer Zeichen-Bibliothek zu beginnen. Die Zeichen 1 bis 31 und 128 bis 255 können mit Hilfe des Programms BILD EINHEIT verwendet werden. Wenn wir ein Programm fertiggestellt haben, verbinden wir es einfach mit BILD EINHEIT. Damit stehen uns alle Zeichen mit einem Male zur Verfügung. Um Speicherplatz zu sparen, können die nicht benötigten Zeichen nachträglich gelöscht werden. Das Anlegen einer Bibliothek in dieser Form erleichtert die wiederholte Benutzung von Zeichen. Es erspart uns den Zeitaufwand, die Zeichen jedesmal neu herstellen zu müssen.

Entwerfen und Verbinden neuer Bilder

Der einfachste Weg zum Entwerfen neuer Bilder besteht darin, diese auf kariertem Papier aufzuzeichnen. Anschließend wird das Bild durch das Eintippen von Zahlen in die DATA-Anweisungen übertragen. Da die Farbpunkte auf dem Bildschirm nicht genau quadratisch sind, wird das Bild auf dem Monitor nicht exakt die gleiche Form haben wie auf der Papiervorlage. In Abbildung 1 ist ein Zwei-Farben-Baum auf kariertem Papier dargestellt.

Verändern Sie BILD EINHEIT wie folgt:

```
10 REM  ..ZWEI FARBEN BAUM-BILD EINHEIT.:
11 :
12 REM  BILD UNTERPROGRAMME + BILD BIBLIOTHEK
13 :
100 GR : HOME
110 X$ = CHR$(1)
120 XC(1) = 4
130 XC(2) = 8
140 GOSUB 15300
999 END
20100 DATA 21,38: REM' ZWEI-FARBEN-BAUM
20102 DATA " 11111111111111"
20104 DATA " 11111111111111"
20106 DATA "111111111111111111"
20108 DATA "111111111111111111"
20110 DATA "11111111111111111111"
20112 DATA "11111111111111111111"
20114 DATA "11111111111111111111"
20116 DATA "11111111111111111111"
20118 DATA "11111111111111111111"
20120 DATA "11111111111111111111"
20122 DATA "11111111111111111111"
20124 DATA "11111111111111111111"
20126 DATA "11111111111111111111"
20128 DATA "11111111 221111111111"
20130 DATA " 11111 22 11111111"
20132 DATA " 11111 22 11111"
20134 DATA " 1111 22 11"
20136 DATA " 22"
20138 DATA " 22"
20140 DATA " 22"
20142 DATA " 22"
20144 DATA " 22"
20146 DATA " 22"
20148 DATA " 22"
20150 DATA " 22"
20152 DATA " 22"
20154 DATA " 22"
20156 DATA " 22"
20158 DATA " 22"
20160 DATA " 22"
20162 DATA " 22"
20164 DATA " 22"
20166 DATA " 22"
20168 DATA " 22"
20170 DATA " 22"
20172 DATA " 22"
20174 DATA " 22"
20176 DATA " 22222222"
20178 DATA "-1"
```

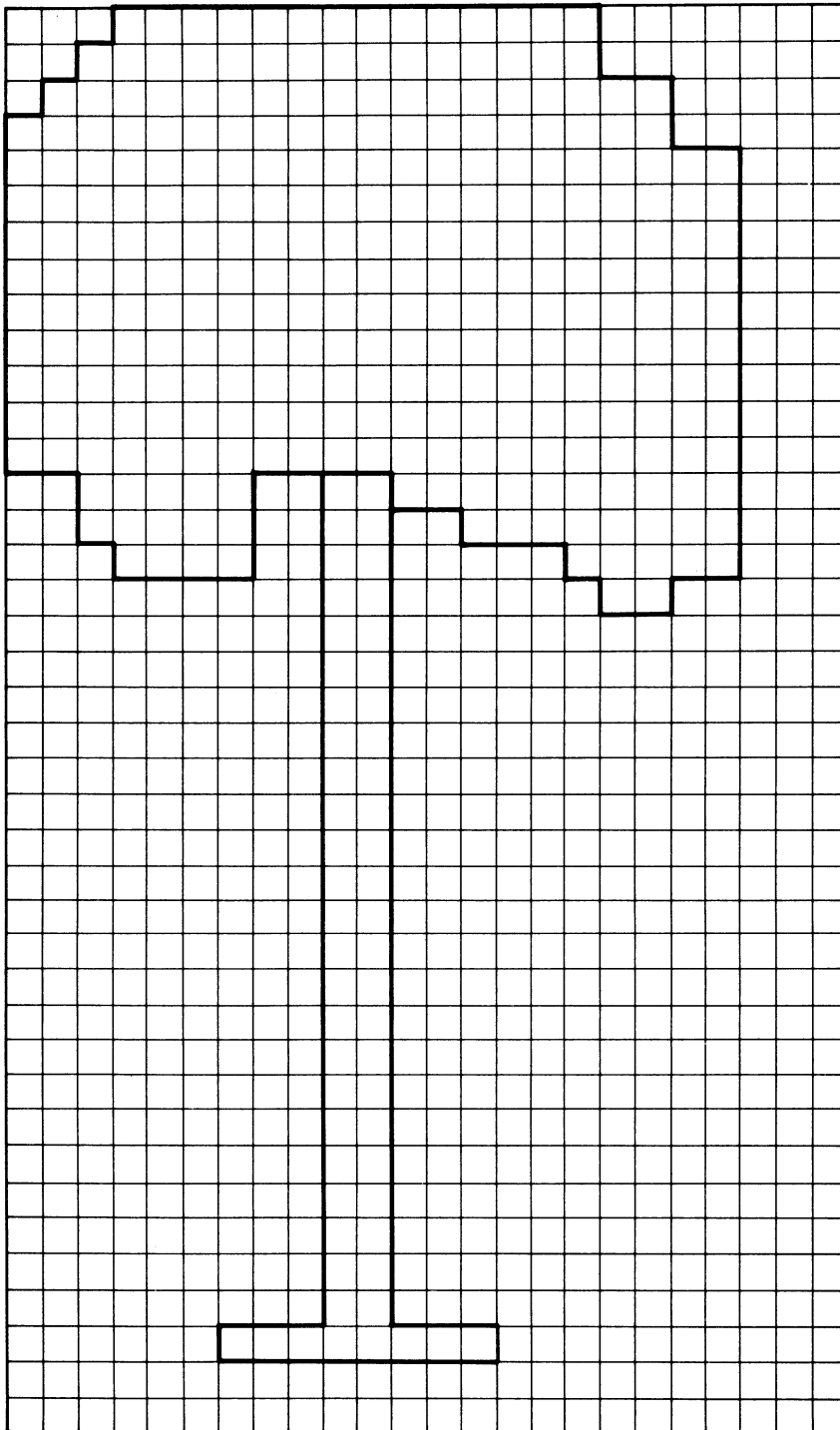


Abb. 1

Die Zeilen 20100 bis 20178 entsprechen dem grafischen Bild. Speichern Sie das Programm als ZWEI-FARBEN-BAUM. Da für den Zwei-Farben-Baum kein entsprechendes ASCII-Zeichen existiert, muß der Zugriff auf dieses Bild auf andere Weise erfolgen (siehe Zeile 110). Anstatt in der Textvariablen X\$ die entsprechenden Zeichen anzugeben, wird die Kennziffer des Bildes benutzt. Um zum Beispiel das Bild Nr. 130 zu spezifizieren, benutzen wir den Befehl: X\$ = CHR\$(130). Mit dieser Methode können wir Bilder auf dem Bildschirm genau so positionieren wie Buchstaben und Zahlen. Wir ordnen die Bildnummer mit CHR\$ der Variablen X\$ zu. Dann erfolgt der Eintritt in das Unterprogramm BILD EINHEIT in Zeile 15300 oder 15400.

Mit X\$ können auch mehrere Bilder in einer Reihe angeordnet werden. Eine solche Zuordnung hat die Form:

X\$ = CHR\$(130) + CHR\$(140).

Die Zahlen 130 und 140 geben die Nummern der Bilder an.

Zum Schluß noch eine Bemerkung zum freien Raum zwischen zwei Bildern. Das Programm BILD EINHEIT setzt eine Leerstelle zwischen zwei aufeinanderfolgende Bilder. Mit XS kann die Anzahl der Leerstellen vor dem Eintritt in das Unterprogramm verändert werden.

Mehrmalige Benutzung eines Bildes mit verschiedenen Farben

Beim Entwerfen des Bildes haben wir jedem Kästchen auf dem Zeichenpapier eine Zahl zugeordnet, die eine bestimmte Farbe repräsentiert. Karos, die keine Zahl enthalten, werden nicht farbig ausgefüllt. Bei der späteren Verwendung im Programm wird jeder Zahl eine APPLE-Farbe zugeordnet. Das geschieht auf die gleiche Art wie mit der Zahl 1 bei den Buchstaben.

Der Zwei-Farben-Baum hat einen braunen Stamm und dunkelgrünes Laub, wenn XC(1) = 4 und XC(2) = 8 gilt. Ein wenig später können wir dasselbe Bild noch einmal benutzen. Diesmal haben wir einen weißen Stamm mit gelben Blättern, indem wir XC(1) = 12 und XC(2) = 15 setzen. Oder wir erhalten einen roten Stamm mit rotem Laubwerk, wenn wir XC(1) = 1 und XC(2) = 1 schreiben.

Durch die Verwendung dieser Farbnumerierungsmethode können wir Bilder entwerfen, ohne direkt entscheiden zu müssen, welche Farbe wir letztendlich verwenden wollen.

Wir können mehrere Farbkombinationen ausprobieren, indem wir einfach verschiedene Farbzusordnungen treffen. Diese Flexibilität ist besonders vorteilhaft, wenn wir Bilder entwerfen wollen, wie zum Beispiel den Baum in Abbildung 2. Für dieses Bild benutzen wir drei Zahlen. Indem wir diesen Zahlen die APPLE-Farben in unterschiedlicher Art und Weise zuordnen, erhalten wir recht abwechslungsreiche Bilder.

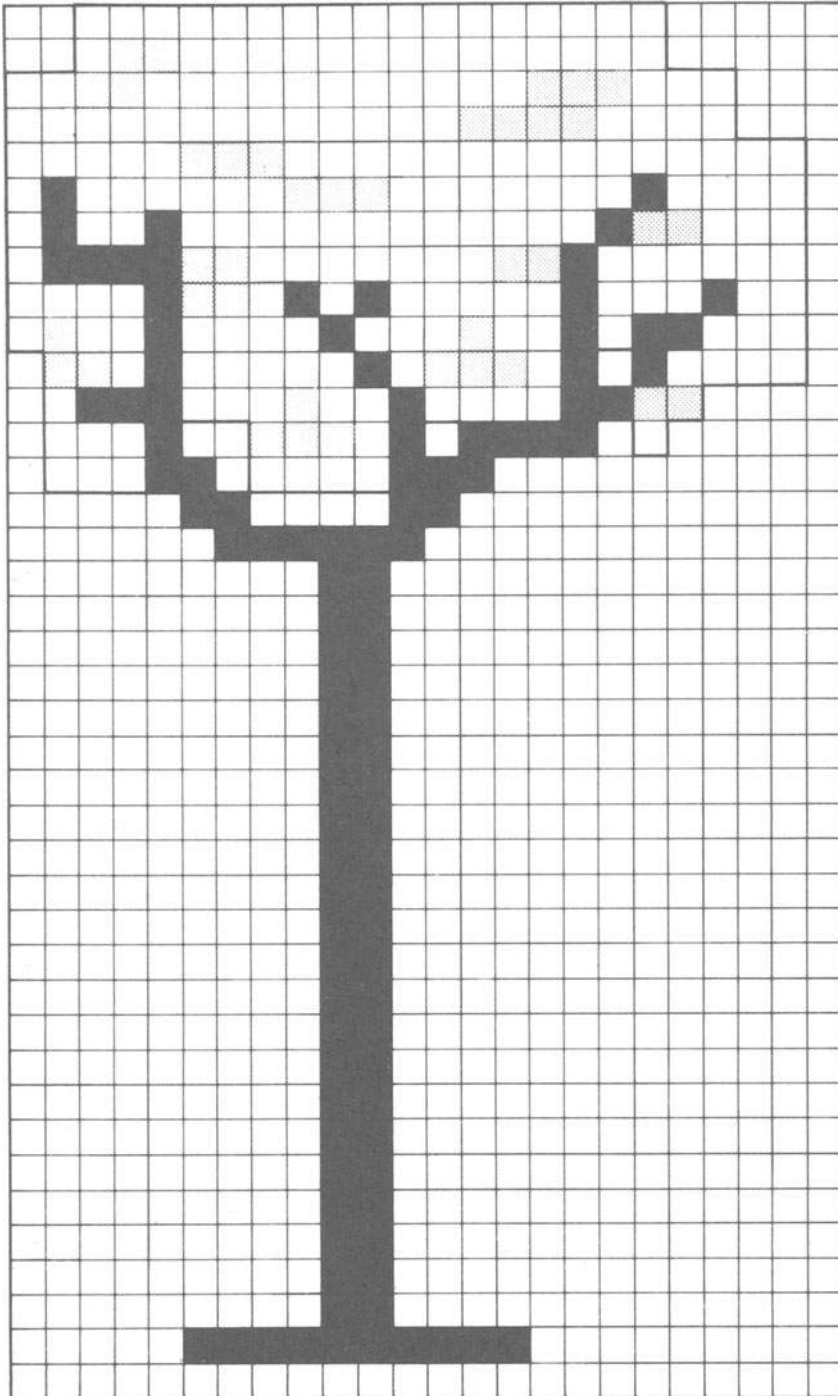


Abb. 2

```

10 REM ...DREI FARBEN BAUM-BILD EINHEIT...
11 :
100 GR : HOME
110 X$ = CHR$ (2)
120 XV = 0
130 XC(1) = 1
140 XC(2) = 2
150 XC(3) = 3
160 GOSUB 15300
170 PRINT : PRINT "DRUECKE EINE TASTE,DANN GEHT'S WEITER.. ";
180 GET Z$
200 GR : HOME
210 XV = 0
220 XC(1) = 3
230 XC(2) = 4
240 XC(3) = 5
250 GOSUB 15300
260 PRINT : PRINT "DRUECKE EINE TASTE,DANN GEHT'S WEITER.. ";
270 GET Z$
300 GR : HOME
310 XV = 0
320 XC(1) = 9
330 XC(2) = 9
340 XC(3) = 9
350 GOSUB 15300
360 PRINT : PRINT "DRUECKE EINE TASTE,DANN GEHT'S WEITER.. ";
370 GET Z$
400 GR : HOME
410 XV = 0
420 XC(1) = 5
430 XC(2) = 5
440 XC(3) = 6
450 GOSUB 15300
460 PRINT : PRINT "DRUECKE EINE TASTE,DANN GEHT'S WEITER.. ";
470 GET Z$
500 GR : HOME
510 XV = 0
520 XC(1) = 0
530 XC(2) = 0
540 XC(3) = 8
550 GOSUB 15300
999 END
20200 DATA 23,39: REM DREI-FARBEN-BAUM
20202 DATA " 1111111111111111"
20204 DATA " 1221111111111111"
20206 DATA "112211111111111222111"
20208 DATA "11111111111122221111"
20210 DATA "1111122211111111111111"
20212 DATA "13111111222111111131111"
20214 DATA "1311311111111111322111"
20216 DATA "1333322111111122311111"
20218 DATA "11113222313111113111311"
20220 DATA "121131111131112113133111"
20222 DATA " 2213111113122213131111"
20224 DATA " 13331112113111113322"
20226 DATA " 1113112221313333 1"
20228 DATA " 1113311111333"
20230 DATA "      33111133"
20232 DATA "      333333"
20234 DATA "      33"
20236 DATA "      33"

```



```

20238 DATA " 33"
20240 DATA " 33"
20242 DATA " 33"
20244 DATA " 33"
20246 DATA " 33"
20248 DATA " 33"
20250 DATA " 33"
20252 DATA " 33"
20254 DATA " 33"
20256 DATA " 33"
20258 DATA " 33"
20260 DATA " 33"
20262 DATA " 33"
20264 DATA " 33"
20266 DATA " 33"
20268 DATA " 33"
20270 DATA " 33"
20272 DATA " 33"
20274 DATA " 33"
20276 DATA " 33"
20278 DATA " 3333333333"
20280 DATA "-1"

```

Speichern Sie dieses Programm als DREI-FARBEN-BAUM.

Wir sehen, daß das gleiche Bild je nach Art der Farbzuoordnung sehr verschiedenartig aussehen kann. (Siehe die Zeilen 130 bis 150, 220 bis 240, 320 bis 340, 420 bis 440 und 520 bis 540).

Als wir den Baum entworfen haben, überlegten wir, welche Bereiche verschiedene Farben haben sollten. Dann ordneten wir jedem Bereich eine Zahl zu. (Je mehr Zahlen benutzt werden, desto größer wird die Vielfalt bei der Farbgebung des Bildes.) Dann benutzten wir dieses Bild im Programm und ordneten jeder Zahl eine Farbe zu. (Wir können auch dieselbe Farbe mehreren Zahlen zuordnen — siehe Zeile 320 bis 340, 420 bis 440 und 520 bis 540.) Mit dieser Methode stehen maximal neun Farben zur Verfügung. Die ständigen Veränderungen der Bilder werden durch aktuelle Farbfestlegungen bewirkt. Natürlich können wir das Bild zu einem späteren Zeitpunkt mit anderen Farben noch einmal verwenden.

Das nächste Bild ist komplexer und vielseitiger als die Vorhergehenden. Wir benutzten neun Zahlen, um die verschiedenen Teile der Figur zu bezeichnen. Durch die Verwendung verschiedener Farbsätze erhalten wir jeweils vier mögliche Arm- bzw. Beinstellungen (Abbildung 3). Den nicht benötigten Körperteilen muß die Farbe schwarz zugeordnet werden.

```

10 REM- ...PERSON-BILD EINHEIT...
11 :
100 C = 3: REM ANZAHL DER FARBEN
110 FOR N = 1 TO C
120 X$ = CHR$(3)
197 :
198 REM * FARBE FUER N'TE PERSON IN ZEILE 500+10*N *
199 :
200 Z = 500 + 10 * N
210 GOSUB 19000

```

```

220 FOR J = 1 TO 9
230 READ XC(J)
240 NEXT
300 GR : HOME
310 XV = 0
320 GOSUB 15300
400 PRINT
410 PRINT "DRUECKE RETURN,DANN GEHT'S WEITER...";
420 GET Z$
430 NEXT
490 END
497 REM * WERTE FUER XC(1),...,XC(9) *
498 REM (FARBEN FUER N'TE PERSON IN ZEILE 500+10*N)
499 :
510 DATA 5,0,0,5,5,0,5,0,5
520 DATA 1,0,1,0,1,0,1,0,1
530 DATA 0,4,4,0,4,4,0,4,0
20300 DATA 20,39: REM PERSON
20302 DATA " 555 333"
20304 DATA " 555 3"
20306 DATA " 555 3"
20308 DATA " 555 3"
20310 DATA " 555 3"
20312 DATA " 5 3"
20314 DATA " 5 3"
20316 DATA " 22222555555553333"
20318 DATA " 2 115555 4"
20320 DATA " 2 11 5555 4"
20322 DATA " 2 11 5555 4"
20324 DATA " 211 5555 4 4"
20326 DATA " 21 5555 4 4"
20328 DATA " 21 5555 444444"
20330 DATA "22211 5555"
20332 DATA " 11 5555"
20334 DATA " 1115555"
20336 DATA " 115555"
20338 DATA " 7777775555"
20340 DATA " 7777775555"
20342 DATA " 77 88 55"
20344 DATA " 77 88 55"
20346 DATA " 77 88 55"
20348 DATA " 77 88 55"
20350 DATA " 77 88 55"
20352 DATA " 77 88 55"
20354 DATA " 77 88 55999999"
20356 DATA "777 88 55999999"
20358 DATA "777 88 66 99"
20360 DATA " 88 66 99"
20362 DATA " 88 66 99"
20364 DATA " 88 66"
20366 DATA " 88 66"
20368 DATA " 88 66"
20370 DATA " 88 66"
20372 DATA " 88 66"
20374 DATA " 88 66"
20376 DATA " 88888 66666"
20378 DATA " 88888 66666"
20380 DATA "-1"

```

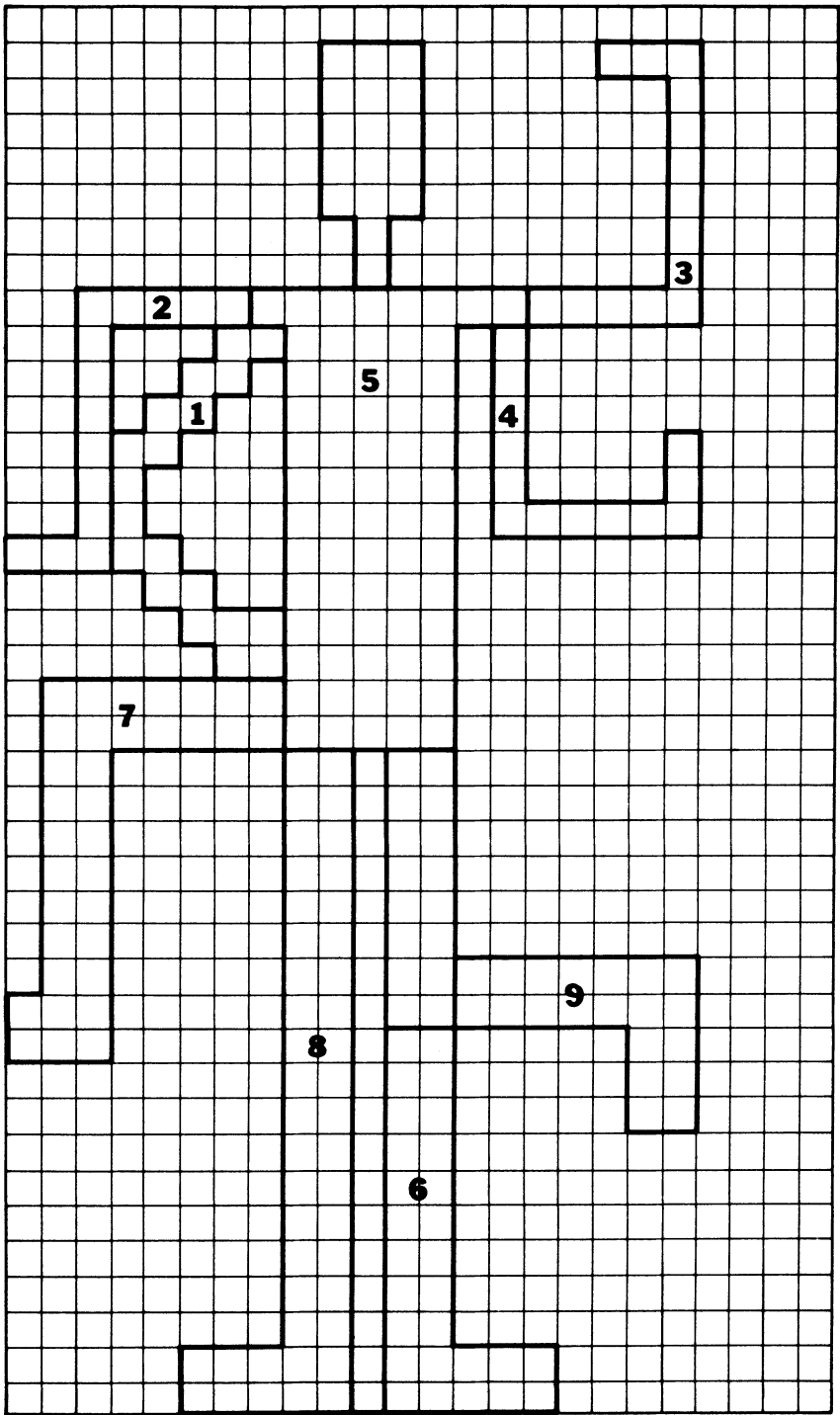


Abb. 3

Benutzen Sie für die Bereiche 1, 3 und 5 dieselbe Farbe und ordnen Sie den Bereichen 2 und 4 die Farbe schwarz zu. Die Figur hat jetzt einen Arm an der Hüfte und streckt den anderen Arm in die Luft.

Soll die Figur gerade stehen, müssen die Bereiche 5, 6 und 8 dieselbe Farbe erhalten. Die Bereiche 7 und 9 werden schwarz gezeichnet.

Können Sie die Farbe so bestimmen, daß die Figur nur den linken Fuß hebt?

Verschiedene Fernsehsysteme, verschiedene Farben

Im letzten Kapitel spielten wir mit Farben bei der Programmierung von Rechtecken und Rändern. In diesem Kapitel erstellten wir Bilder. Hier wird der Leser sicherlich viel Wert auf die gute Darstellung der Farben legen. Die Farben, die wir auf dem Monitor sehen, hängen jedoch stark von dem speziellen Fernsehgerät ab. So kann in der Tat ein Bild ganz anders aussehen, wenn es auf einem anderen Fernsehgerät dargestellt wird. Die Firma APPLE liefert dazu eine Testkarte für Farben. Mit dieser Karte (sie befindet sich in der APPLE Bedienungsanleitung) können wir sehen, wie die Farben aussehen sollten.

Wir möchten dem Leser empfehlen, eigene Bilder zu entwerfen. Es macht sehr viel Spaß und gibt den Programmen einen individuellen Charakter. Große und aufwendige Bilder können innerhalb anderer Programme erscheinen. Oder wir stellen ein kleines Bild her und lassen es um den Bildschirm wandern. Es ist wichtig, sich zu merken, daß alle Bilder in den DATA-Anweisungen enthalten sind. Deshalb können nicht dieselben Zeilennummern zur Darstellung verschiedener Bilder benutzt werden.

Zusammenfassung von Kapitel 3

Wir haben gezeigt, wie das Programm BILD EINHEIT zur Darstellung von Buchstaben und Zahlen in LO-RES-Grafik benutzt wird. Außerdem haben wir gesehen, wie hübsche und originelle Bilder entworfen werden. Das Programm BILD EINHEIT ist beim Schreiben von eigenen Spielprogrammen sehr nützlich. An anderer Stelle in diesem Buch werden wir zeigen, wie wir dieses Programm in unseren Spielen eingesetzt haben.

Lösungen zu Kapitel 3

Frage 1: Das Wort "KATZE" erscheint in violetter Schrift auf dem Bildschirm. Anschließend wird dasselbe Wort an gleicher Stelle in gelber Farbe ausgedruckt.

Frage 2: Der Bildschirm wird gelöscht, bevor das Wort "KATZE" in gelber Farbe ausgedruckt wird.

Kapitel 4

Hochauflösende Grafik

In diesem Kapitel wollen wir etwas über die Nutzung der hochauflösenden Grafik des APPLE II lernen. Im Vergleich zur LO-RES-Grafik ist die hochauflösende Grafik (HI-RES) wesentlich komplizierter anzuwenden. Deshalb wollen wir nicht alle Einzelheiten zeigen, sondern uns auf einige wenige Dinge beschränken, die das Programmieren der HI-RES-Grafik in APPLESOFT BASIC ohne großen Aufwand erlauben. Um die Darstellung der Bilder auf dem Bildschirm zu beschleunigen, benutzen viele Spielprogramme HI-RES-Grafik mit Unterprogrammen in Maschinensprache. Da diese nicht in BASIC geschrieben sind und da Maschinensprachen über das Ziel dieses Buches hinausgehen, wollen wir hier nur so viel sagen, daß die HI-RES-Programme wesentlich komplizierter sind als sie auf den ersten Blick erscheinen.

Die Konstrukteure des APPLE versuchten die Schwierigkeiten beim Programmieren von HI-RES-Bildern in BASIC durch das Einführen von Formativtabellen auszuräumen. Mittels Formativtabellen können grafische Motive entworfen, hergestellt und wieder verändert werden. Dazu gibt es einen speziellen Befehlssatz. Doch leider sind auch die Formativtabellen sehr umständlich und schwierig zu handhaben. Sie sind außerdem zu kompliziert, um in diesem Buch näher erläutert zu werden. Auch bleibt die Darstellung der Grafik mit diesen Formativtabellen weiterhin langsam. Sie sind zum Beispiel nicht dazu geeignet, eine HI-RES-Karte von Europa zu erstellen. Für diejenigen Leser, die nähere Einzelheiten über die HI-RES-Formativtabellen wissen möchten, verweisen wir auf Kapitel 9 der APPLESOFT-Programmieranleitung. Außerdem ist es empfehlenswert, eines der im Handel erhältlichen Software-Pakete für High-Resolution-Grafik zu erwerben; sie enthalten meist eine ausführliche Programmdokumentation. Die Benutzung eines kommerziellen Programms ist viel einfacher als die Programmierung von HI-RES-Grafik in BASIC. Sehen Sie sich bei einigen der Software-Anbieter um und studieren Sie genau die Programmbeschreibungen, damit Sie die Anwendung der Programme gut verstehen.

Grundlagen der HI-RES-Grafik

Da wir jetzt wissen, was in BASIC nicht machbar ist, zeigen wir einige leicht durchführbare Anwendungen. HI-RES arbeitet mit zwei Grafikseiten: Seite 1 und Seite 2. Um Seite 1 zu benutzen, verwenden wir den Befehl HGR. Mit HGR2 wird der Computer angewiesen, Seite 2 zu benutzen.

Jeder dieser Befehle löscht die zugehörige Seite mit schwarzer Farbe. Mit Grafikseite 1 kann ein Raster von 280 (0–279) Punkten Breite und 160 (0–159) Punkten Höhe dargestellt werden. Unterhalb der Grafikseite 1 steht noch ein nicht benutzter Raum zum Ausdruck von vier Textzeilen zur Verfügung. Dafür wird im BASIC-Programm der übliche PRINT-Befehl verwendet. Die Grafikseite 2 verfügt über ein Raster von 280 × 192 Punkten. Sie bietet keinen Platz für Textzeilen.

Der Befehl HCOLOR bestimmt die Farbe beim Ausdrucken in HI-RES-Grafik. Es stehen 8 Farben zur Verfügung. Ihre Kennzahlen lauten wie folgt:

0 = SCHWARZ	4 = SCHWARZ
1 = GRÜN	5 = ORANGE
2 = VIOLETT	6 = BLAU
3 = WEISS	7 = WEISS

Die Farben Schwarz und Weiß sind zweimal vertreten (die Gründe hierfür sind technischer Art). Möglicherweise erscheinen die Farben 5 und 6 auf Ihrem Bildschirm nicht als Orange bzw. Blau. Eine der Schwierigkeiten der HI-RES-Farben liegt in den unterschiedlichen Fernsehsystemen; die Darstellung der Farben ist von der Verwendung des speziellen Fernsehgeräts abhängig. Um einige dieser Probleme zu vermeiden, kann man in Schwarz-Weiß programmieren.

Um die Farbe Weiß zu erhalten, benutzen wir den Befehl:

```
HCOLOR = 3
```

Die Farbe wird solange verwendet, bis ein weiterer HCOLOR-Befehl eine neue Farbe bestimmt.

Der HPLOT-Befehl bewirkt das Zeichnen eines Punktes oder einer Linie auf der HI-RES-Seite. Die obere linke Ecke entspricht dem Punkt 0,0. Alle Punkte werden in Relation zu diesem Punkt ausgedruckt.

Der Befehl HPLOT 25,55 bewirkt das Ausdrucken eines Punktes in der 25. Spalte von links und der 55. Zeile von oben. Dies geschieht in der zuletzt benutzten Farbe.

Mit HPLOT 10,20 TO 110,90 ergibt sich eine diagonale Linie von Punkt 10,20 zu Punkt 110,90. Um die Linie über Punkt 110,90 hinaus fortzusetzen, benutzen wir die verkürzte Form des HPLOT-Befehls:

```
HPLOT TO 160,20
```

Die verkürzte Form des HPLOT-Befehls setzt voraus, daß die Linie vom letzten ausgedruckten Punkt an fortgesetzt werden soll (in unserem Fall 110,90).

Die obigen Befehle können in einer einzigen Anweisung zusammengefaßt werden:

HPlot 10,20 TO 110,90 TO 160,20

Das folgende Programm demonstriert die Anwendung der bisher erläuterten HI-RES-Befehle:

```
10 REM ...HI-RES DEMO1...
11 :
110 HGR
120 HCOLOR= 3
130 HPlot 25,55
140 GOSUB 220
150 HPlot 10,20 TO 110,90
160 GOSUB 220
170 HPlot TO 160,20
180 GOSUB 220
190 GOTO 300
215 :
220 PRINT "MIT 'RETURN' GEHT ES WEITER...";
230 INPUT R$
240 RETURN
300 END
```

Geben Sie dieses Programm ein und starten Sie es.

Frage 1: Welcher Befehl bewirkt das Ausdrucken einer senkrechten Linie von Punkt 160,20 zu Zeile 90?

Schreiben Sie diesen Befehl in Programmzeile 185.

Frage 2: Schreiben Sie in Zeile 190 einen Befehl, der den Ausdruck in grüner Farbe bewirkt. Wie erhalten Sie eine waagerechte Linie über den ganzen Bildschirm am oberen Bildrand?

Schreiben Sie diesen Befehl in Zeile 200.

Die Befehle HGR und HGR2 löschen den Bildschirm mit schwarzer Farbe. Um den Bildschirm mit farbigem anstatt schwarzem Hintergrund zu löschen, verwenden wir folgende Befehle:

- HGR oder HGR2
- POKE 28,X
- CALL 62454

Die Variable X entspricht einer Zahl zwischen 0 und 255. Für die folgenden Werte von X bewirkt der POKE-Befehl ein Löschen des Bildschirms in den Farben:

SCHWARZ	0 oder 128
WEISS	127 oder 255
GRÜN	42
VIOLETT	85
ORANGE	170
BLAU	213

Für jeden anderen Zahlenwert erhalten wir ein interessantes Streifenmuster auf dem Bildschirm. Das Löschen des Schirms erfolgt sehr schnell. Ist die Hintergrundfarbe festgelegt, können wir mit anderen Farben Punkte und Linien über den Hintergrund zeichnen lassen.

Einige HI-RES-Probleme

Unser letztes Programm funktioniert genau wie erwartet. Auch wenn wir die Farbe ändern, wird das Programm weiter richtig arbeiten. Dies ist auch nach dem Umschalten auf Grafikseite 2 der Fall. Mit Ausnahme des Textes "DRUECKE RETURN" arbeitet das Programm wie erwartet. Wegen des fehlenden Textfensters können diese Worte nicht erscheinen.

Daß das Programm genau so funktioniert, wie wir das erwartet haben, erwähnen wir deshalb ausdrücklich, weil das nicht immer der Fall ist. Die Unterschiede in den Fernsehsystemen und Besonderheiten der HI-RES-Grafik bewirken, daß wir nicht immer das sehen, was zu erwarten ist. Dazu zeigen wir nun ein ‚klassisches‘ Beispiel. Das folgende Programm versucht, einen schönen, einfarbigen Rand in der Grafikseite 1 zu zeichnen. Geben Sie das Programm ein und starten Sie es. Was geschieht?

```
10 REM    ...HIRES UMRANDUNG...
11 :
12 REM    DEMONSTRIERT UMRANDUNG IN JEDER FARBE
13 :
100 FOR J = 0 TO 7: REM    VERWENDE ALLE FARBEN
110 HOME : HGR
120 HCOLOR= J: REM    NAECHSTE FARBE
200 HPLOT 0,0 TO 279,0 TO 279,159 TO 0,159 TO 0,0
300 VTAB 22: PRINT "UMRANDUNG IN FARBE ";J
320 PRINT "DRUECKE RETURN FUER DIE NAECHSTE FARBE"
325 PRINT "PRESS ESCAPE TO STOP...";
330 GET Z$
340 IF Z$ = CHR$(27) THEN TEXT : END : REM    ESC
350 NEXT
360 GOTO 100: REM    SCHLEIFE
```

Wir sehen, daß einige der Ränder nicht vollständig sind. Andere Linien erscheinen in mehreren Farben. Wie kommen diese Effekte zustande? Die Antwort auf diese Frage ist nicht einfach.

Frage 3: Warum erscheint bei den Farben 0 und 4 nichts auf dem Bildschirm?

Frage 4: Welche Farben ergeben einen vollständigen Rand, obwohl mehrere Farben gemischt werden?

Fügen Sie den folgenden Befehl zum Programm hinzu und starten Sie es erneut. Beobachten Sie die Veränderungen.

```
210 HPLOT 1,1 TO 278,1 TO 278,158 TO 1,158 TO 1,1
```


Mit diesem Befehl wird ein doppelter Rand erzeugt, indem ein weiterer innerer Rand hinzugefügt wird. Wir wollen feststellen, ob dadurch der ursprüngliche Rand verbessert wird.

Frage 5: Welche Farben ergeben nun einen vollständigen, einfarbigen Rand?

Die Farbe 5 (Orange) erscheint weiter in zwei Farben. Auch die Farbe Weiß wird nicht korrekt wiedergegeben. Es ist überraschend, daß dieses Problem auch dann besteht, wenn ein teurer und hochwertiger Video-Monitor verwendet wird.

Streichen Sie Zeile 200 und schauen Sie, ob der innere Rand allein richtig wiedergegeben wird.

Frage 6: Was geschieht, wenn das Programm jetzt gestartet wird?

Das gleiche Problem taucht auch bei Benutzung von Grafikseite 2 auf. Geben Sie dieses kleine Programm ein und starten Sie es:

```
100 HGR
110 HCOLOR = 1
120 HPLOT 50,0 TO 70,150
```

Frage 7: Wir erwarten, daß auf dem Bildschirm eine einfache Linie ausgedruckt wird. Was geschieht tatsächlich?

Wählen Sie eine andere Farbe. Schauen Sie, ob das Bild sich dadurch ändert! Wählen Sie andere Bildschirmkoordinaten für den Ausdruck der Linie. Das Bild auf dem Monitor ändert sich in einer interessanten Form.

Es gibt noch weitere Probleme mit der HI-RES-Grafik. Einige Linien führen zu einem verwirrenden Durcheinander von Farben. Ein für alle Fernsehsysteme gemeinsames Problem besteht in einer unerwünschten orangen Linie am linken Bildschirmrand. Dies ist eine Folge der Konstruktion der HI-RES-Software. Andere Probleme hängen nicht von den Grafikfähigkeiten des APPLE-Computers ab. Allerdings sind sie ein Produkt elektronischer Schaltungen in Fernsehgeräten und teuren Farbmonitoren.

Spaß mit HI-RES

Wir wollen dieses Kapitel nicht beenden, ohne einige Beispiele vorzuführen, die leicht und schnell mit der HI-RES-Grafik durchführbar sind. Wir können vieles mit HI-RES realisieren, nur sollten wir keine Vollkommenheit erwarten. Beim Durchführen der folgenden Übungen werden einige farbliche Mängel auf dem Bildschirm sichtbar. Aber nicht verzweifeln ... das gehört eben mit zur HI-RES-Grafik hinzu!

Geben Sie das folgende Programm ein. Es zeigt ein einfaches Gittermuster in HI-RES-Grafik:

```

10 REM    ...GITTER NETZ...
11 :
12 REM    EINFACHES GITTER NETZ
13 :
100 TEXT : REM    BENUTZE VOLLEN BILDSCHIRM
110 H0 = 0: REM    H-URSPRUNG
120 V0 = 159: REM    V-URSPRUNG
130 MS = 19: REM    MAXIMALE SCHRITTWEITE
200 HOME : HGR
210 S% = MS * RND (1) + 1: REM    WAEHLE SCHRITTWEITE DURCH ZUFALLSFUNKTI
    ON
220 HTAB 1: VTAB 22: PRINT "SCHRITTWEITE = ";S%
230 Z% = 7 * RND (1) + 1: IF Z% = 4 THEN 230: REM    WAEHLE ZUFAELLIGE NI
    CHT-SCHWARZE FARBE
240 HCOLOR= Z%
300 R% = V0 / S% * S%: REM    INTERVALL
310 FOR J = 0 TO R% STEP S%: REM    SCHRIITE INNERHALB DER INTERVALLGRENZ
    EN
320 HPLOT H0,V0 - R% + J TO H0 + J,V0: REM    NAECHSTES GITTER SEGMENT
390 NEXT
510 PRINT "DRUECKE RETURN FUER DAS NAECHSTE GITTER"
515 PRINT "PRESS ESCAPE TO STOP...";
520 GET Z$
530 IF Z$ = CHR$ (27) THEN TEXT : END : REM    ESC
540 GOTO 200

```

Speichern Sie es unter dem Namen GITTER NETZ ab.

Nun zeigen wir kunstvollere Versionen eines geometrischen Gittermusters, die sehr einfach herzustellen sind und klar und deutlich auf dem Bildschirm erscheinen. Unser spezielles Beispiel ist etwa komplizierter und ruft bei manchen Mustern künstliche Farbveränderungen hervor. Ändern Sie GITTER NETZ wie folgt und speichern Sie das neue Programm als GITTER NETZ2.

```

10 REM    ...GITTER NETZ2 - GITTER NETZ
11 :
12 REM    SCHOENERES GITTER NETZ
13 :
110 H0 = 139: REM    H-URSPRUNG
120 V0 = 79: REM    V-URSPRUNG
130 MS = 11: REM    MAXIMALE SCHRITT WEITE
330 HPLOT TO H0,V0 + R% - J: REM    UNTERE RECHTE ECKE
340 HPLOT TO H0 - J,V0: REM    UNTERE LINKE ECKE
350 HPLOT TO H0,V0 - R% + J: REM    OBERE LINKE ECKE

```

Starten Sie das Programm. Beobachten Sie den Bildschirm. Was geschieht? Wir können jedes dieser zwei Programme mit anderen Spielprogrammen verbinden. Dies gibt uns die Möglichkeit, eine Unterbrechung des Spiels effektiv anzuzeigen bzw. ein interessantes Schlußmotiv am Ende des Spiels darzustellen.

Zusammenfassung von Kapitel 4

Vielleicht ist dieses Kapitel für diejenigen Leser enttäuschend gewesen, die in wenigen Seiten alle Tricks der HI-RES-Grafik kennenlernen wollten. Wir glauben, daß die Programmierung von HI-RES-Grafiken über die Möglichkeiten des durchschnittlichen Hobby-Programmierers zu Hause oder in der Schule hinausgeht. Das Programmieren mit HI-RES-Grafik macht mehr Spaß, wenn der Leser ein kommerzielles Software-Paket erwirbt und benutzt. Außerdem möchten wir daran erinnern, daß die LO-RES-Grafik sehr viel einfacher zu gebrauchen ist. Kinder finden LO-RES-Bilder genauso schön wie HI-RES-Bilder.

Lösungen zu Kapitel 4

Frage 1: 185 HPLOT TO 160,90

Frage 2: 190 HCOLOR = 1
200 HPLOT 0,90 TO 279,90

Frage 3: Dies sind schwarze Farben. Sie sind auf einem schwarzen Bildschirm nicht sichtbar.

Frage 4: Die Farben 3 und 7 sind weiß. Mit diesen Farben erscheinen alle vier Seiten der Umrandung, obwohl die vertikalen Seiten ungerade Farbkoordinaten haben.

Frage 5: Grün, Violett, Weiß und Blau werden normal wiedergegeben.

Frage 6: Wie beim ersten Programmdurchlauf taucht das Problem der ungeraden Farben wieder auf.

Frage 7: Anstatt einer durchgehenden Linie erscheint eine Folge von kurzen Linien, die von Punkt zu Punkt verlaufen.

Kapitel 5

Programme zur Dateneingabe

In diesem Kapitel stellen wir einige Unterprogramme vor, die speziell für die Dateneingabe bei Computerspielen entwickelt wurden. Es sind folgende Programme:

- das „Universelle Eingabe“-Programm
- das „Zahlen Eingabe“-Programm
- das „J/N“-Programm
- das „Einzelne Zeichen Eingabe“-Programm
- das „Pause oder Tastendruck Eingabe“-Programm
- das „Tastendruck ohne Wiedergabe“-Programm.

Sicherlich möchten viele interessierte Leser nähere Einzelheiten zu den Dateneinlese-Prozeduren wissen. Deshalb sind einige Teile des Kapitels mehr technischer Art. Für den Leser, der es nicht so genau wissen möchte, genügt es, die anwendungsbezogenen Teile des Kapitels zu lesen.

Ein grundsätzliches Ärgernis für Benutzer von Computerspielen ist der Abbruch eines Programmes mitten im Spielverlauf aufgrund der Eingabe falscher Daten. Umgekehrt gilt dies auch für den Programmierer eines Computerspiels, z.B. wenn unerfahrene Spieler falsche Daten eingeben oder falsche Tasten betätigen. Wie schon erwähnt, führt dies meistens zum Programmabbruch: zur gegenseitigen Enttäuschung des Spielers als auch des Programmierers.

Unser endgültiges Programm enthält Prozeduren zur Bearbeitung von Dateneingaben, testet die Gültigkeit der Daten und antwortet entsprechend. Der Abbruch eines Programmes wird in jedem Falle vermieden. Ein gutes Programm zur Dateneingabe ist immer mit Rücksicht auf den unerfahrenen Benutzer konstruiert. Es wird deshalb nur auf die vorgesehenen Tasten ansprechen. Der Rest der Tastatur wird ignoriert. Wir haben vier Unterprogramme zur Dateneingabe entwickelt, die dieser Beschreibung entsprechen. Außerdem bieten wir zwei weitere Unterprogramme an, die für spezielle Zwecke benutzt werden können.

Hier ist das komplette Programm EINGABE EINHEIT. Es enthält alle weiter oben aufgezählten Unterprogramme.

```

10 REM   ...EINGABE EINHEIT...
11 :
12 REM   EINGABE DER UNTERPROGRAMME
13 :
9990 :
9991 :
9992 REM   ** EINGABE UND WIEDERHOLUNG EINES TEXTES MIT RETURN **
9993 REM   EINTRITT:CURSOR WIRD ZUM BEGINN DES EINGABE-FELDES GESETZT
9994 REM           YW   FELD LAENGE
9995 REM           YF$  PLATZHALTER FUER SPAETERE ZEICHEN
9996 REM   AUSGANG: Z$  TEXT
9997 REM           Z%   -1 (ESC); 0 (NICHT ESC)
9999 :
10000 IF LEN (YF$) < > 1 THEN YF$ = " ": REM   INITIALISIERE YF$ WENN NOTW
      ENDIG
10010 YH% = PEEK (36) + 1: REM   H-POS
10020 YV% = PEEK (37) + 1: REM   V-POS
10100 GOSUB 10500: REM   SETZE DAS EINGABE FELD ENTSPRECHEND YF$ UND INITIALI
      SIERE
10110 GET Z1$
10120 IF Z1$ = CHR$ (13) THEN RETURN
10130 IF Z1$ < > CHR$ (27) THEN 10200: REM   ESC
10140 GOSUB 10500
10150 FLASH : PRINT "ESC"; CHR$ (8);: NORMAL
10160 Z% = - 1: REM   ESC FLAG
10170 GOTO 10110
10200 IF Z1$ < > CHR$ (8) THEN 10300: REM   LINKER PFEIL
10210 IF Z% = - 1 OR LEN (Z%) < = 1 THEN 10100: REM   ABFRAGE ESC ODER EIN
      ZEICHEN ODER WENIGER
10220 PRINT CHR$ (8);YF$; CHR$ (8);: REM   LOESCHE EIN ZEICHEN
10230 Z$ = LEFT$ (Z$, LEN (Z%) - 1)
10240 GOTO 10110
10300 IF Z1$ < " " THEN 10110: REM   IGNORIERE ANDERE KONTROLLZEICHEN
10310 IF Z% = - 1 THEN GOSUB 10500: REM   LOESCHE ESCAPE BEDINGUNG
10320 IF LEN (Z%) < YW THEN 10400
10330 IF YW = 0 THEN 10110: REM   WIEDERHOLE NICHT WENN LAENGE =0
10340 PRINT CHR$ (8);: REM   BEREITS BEI MAXIMALER LAENGE
10350 IF LEN (Z%) = 1 THEN Z$ = ""
10360 IF LEN (Z%) > 1 THEN Z$ = LEFT$ (Z$, LEN (Z%) - 1)
10400 PRINT Z1$;: REM   WIEDERHOLE UND FUEGE ZEICHEN HINZU
10410 Z$ = Z$ + Z1$
10420 GOTO 10110
10500 HTAB YH%; VTAB YV%; FOR Z = 1 TO YW: PRINT YF$;: NEXT : REM   BESCHREI
      BE FELD ENTSPRECHEND YF$
10510 PRINT " ": REM   UND LOESCHE MOEGELICHEN CURSOR
10520 IF YW < 2 THEN FOR Z = YW + 1 TO 3: PRINT " ": NEXT : REM   LOESCHE M
      OEGELICHES ESC WENN FELD NICHT LANG GENUG
10530 HTAB YH%; VTAB YV%
10540 Z$ = ""
10550 Z% = 0
10560 RETURN
10591 :
10592 REM   * EINGABE ZAHL *
10593 REM   EINGANG: BEDINGUNGEN FUER EINGABE TEXTSATZ
10594 REM   AUSGANG: Z% -1 (ESC); 0 (UNGUELTIG); 1 (GANZE ZAHL); 2 (DEZIMALZ
      AHL)

```

```

10595 REM          Z WERT (WENN GUELTIG)
10599 :
10600 GOSUB 10000: REM  LESE TEXT
10610 IF Z% = - 1 OR LEN (Z%) = 0 THEN RETURN : REM  NUR ESC ODER RETURN(
      Z%=0)
10620 Z% = 1: REM  SETZE GUELTIGES ZEICHEN
10630 FOR Z1 = 1 TO LEN (Z%):Z1% = MID% (Z%,Z1,1)
10640 IF Z1% = "." AND Z% = 1 THEN Z% = 2: GOTO 10660: REM  BESTIMMUNG DES E
      RSTEN DEZIMALPUNKTES
10650 IF (Z1% < "0" OR Z1% > "9") AND (Z1% < > "-" AND Z1 > 1) THEN Z% = 0: REM
      UNGUELTIG WENN KEINE ZAHL ODER EIN "-" ZEICHEN
10660 NEXT
10670 Z = VAL (Z%): REM  WERT NUR BEI GUELTIGER FLAG (Z%=1 OR 2)
10680 RETURN
10991 :
10992 REM  ** EINGABE GANZE ZAHL **
10993 REM  EINGANG:BEDINGUNGEN FUER EINGABE TEXT SATZ
10994 REM          YL MINIMALE GANZE ZAHL
10995 REM          YH MAXIMALE GANZE ZAHL
10996 REM  AUSGANG: Z% -1 (ESC); 0 (UNGUELTIGE AZHL); 1 (GUELTIGE ZAHL)
10997 REM          Z -WERT (WENN GANZE ZAHL GUELTIG)
10999 :
11000 GOSUB 10600: REM  EINGABE ZAHL
11010 IF Z% < 1 THEN RETURN : REM  ESC ODER UNGUELTIG
11020 IF Z% = 2 THEN Z% = 0: RETURN : REM  UNGUELTIG WENN DEZIMALPUNKT
11030 IF Z < YL OR Z > YH THEN Z% = 0: REM  UNGUELTIG WENN AUSSERHALB DER IN
      TERVALLGRENZEN
11040 RETURN
11091 :
11092 REM  ** EINGABE DEZIMALZAHL **
11093 REM  EINGANG:BEDINGUNGEN FUER EINGABE TEXTSATZ
11094 REM          YL MINIMALER WERT
11095 REM          YH MAXIMALER WERT
11096 REM  AUSGANG: Z% -1 (ESC); 0 (UNGUELTIG); 1 (GANZE ZAHL); 2 (DEZIMALZ
      AHL)
11097 REM          Z-WERT (WENN GUELTIG)
11099 :
11100 GOSUB 10600: REM  EINGABE ZAHL
11110 IF Z% < 1 THEN RETURN : REM  ESC ODER UNGUELTIG
11120 IF Z < YL OR Z > YH THEN Z% = 0: REM  UNGUELTIG WENN AUSSERHALB DER IN
      TERVALLGRENZEN
11130 RETURN
11191 :
11192 REM  ** EINGABE J ODER N **
11193 REM  EINGANG:CURSOR UND YF% GESETZT
11194 REM  AUSGANG: Z% -1 (ESC); 0 (WEDER J NOCH N ); 1 (J), 2 (N)
11199 :
11200 Y% = "JN": REM  BENUTZE UNTERPROGRAMM ZUR EINGABE EINES EINZELNEN ZEIC
      HENS
11291 :
11292 REM  ** EINGABE EINES EINZELNEN ZEICHEN UND VERBINDEUNG MIT GUELTIGEM
      TEXT **
11293 REM  EINGANG:CURSOR UND YF% GESETZT
11294 REM          Y% TEXT VON VERBINDUNGSZEICHEN
11295 REM  AUSGANG: Z% -1 (ESC); 0 (ZEICHEN NICHT IM TEXTSATZ); J (J'TES ZE
      ICHEN IM VERBINDUNGSTEXT)
11299 :
11300 YW = 1: REM  SETZE FELD-LAENGE
11310 GOSUB 10000
11320 IF Z% = - 1 OR LEN (Z%) = 0 THEN RETURN : REM  NUR ESC ODER RETURN(
      Z%=0)

```

```

11330 Z% = 0: REM   SETZE ZEICHEN FUER NICHT VERBUNDEN
11340 FOR Z1 = 1 TO LEN (Y%)
11350 IF Z% = MID% (Y%,Z1,1) THEN Z% = Z1: REM   VERBINDE MIT POSITION Z1
11360 NEXT
11370 RETURN
11391 :
11392 REM   ** PAUSE FESTE LAENGE ODER BIS TASTENDRUCK **
11393 REM   EINGANG:YP   LAENGE DER PAUSE IN INTERNEN ZEITEINHEITEN
11394 REM               0   WARTET AUF TASTENDRUCK
11395 REM   AUSGANG: Z%  -1 (ESC); 0 (PAUSE ABGELAUFEN); 1 (TASTENDRUCK BEVO
      R PAUSE ABGELAUFEN)
11396 REM               Z   TASTENDRUCK (ASCII-WERT  + 128)
11399 :
11400 POKE  - 16368,0: REM   LOESCHE BISHERIGES ZEICHEN
11410 Z1 = 0: REM   INITIALISIERE ZAEHLER (* EINGANG FUER EINZELNE TASTEN*)
11420 Z1 = Z1 + 1
11430 Z = PEEK ( - 16384)
11440 IF Z > = 128 THEN Z% = 1 - 2 * (Z = 155): RETURN : REM   TASTENDRUCK; A
      BFRAGE FUER ESC DANN RETURN
11450 IF Z1 < YP OR YP = 0 THEN 11420
11460 Z% = 0: REM   PAUSE ABGELAUFEN
11470 RETURN
11491 :
11492 REM   ** EINGABE EINZELNER ZEICHEN, KEINE WIEDERHOLUNG, OHNE VORANGEHE
      NDES ZEICHEN **
11493 REM   AUSGANG: Z%  -1 (ESC); 1 (ANDERE TASTE)
11494 REM               Z   TASTENDRUCK (ASCII-WERT  + 128)
11499 :
11500 YP = 0: GOSUB 11400: REM   WARTET AUF TASTENDRUCK
11510 POKE  - 16368,0: RETURN : REM   LOESCHE TASTENFUNKTIONEN UND RETURN
11591 :
11592 REM   ** EINGABE EINZELNER ZEICHEN, KEINE WIEDERHOLUNG, MIT VORANGEHEN
      DEM ZEICHEN **
11593 REM   AUSGANG: Z%  -1 (ESC); 1 (ANDERE TASTE)
11594 REM               Z   TASTENDRUCK (ASCII-WERT  + 128)
11599 :
11600 YP = 0: GOSUB 11420: REM   EINGABE EINER TASTE, KEIN ZEICHEN VORHER
11610 POKE  - 16368,0: RETURN : REM   LOESCHE TASTENFUNKTIONEN UND RETURN

```

Das Unterprogramm „Universelle Eingabe“

Das Unterprogramm zur universellen Dateneingabe akzeptiert alle Zeichen der Tastatur: Zahl, Buchstabe und spezielle Symbole. Es kann zur Eingabe aller Daten benutzt werden. Wir dagegen benutzen es nur für die Eingabe von Zahlen und speziellen Symbolen. Das Unterprogramm simuliert die Verwendung des üblichen INPUT-Befehls in BASIC. Dies setzt voraus, daß der Benutzer die RETURN-Taste betätigt, um anzuzeigen, daß die Dateneingabe abgeschlossen ist. Einige Programmierer verwenden sowohl GET- als auch INPUT-Befehle in einem Programm. Es ist für ungeübte Benutzer sehr verwirrend, für manche Antworten die RETURN-Taste drücken zu müssen und für andere nicht. Bei unserem Programm ist bei jeder Dateneingabe die RETURN-Taste erforderlich. (Ein technischer Hinweis: Zwar wird für die Dateneingabe eine GET-Anweisung benutzt, aber jede Eingabe wird auf das RETURN-Zeichen überprüft. Die Eingabe-Prozedur endet erst mit Eingabe des RETURN-Zeichens.)

Eine weitere Programmvereinbarung gibt uns die Möglichkeit, zu jeder Zeit die ESCAPE (ESC)-Taste zu drücken. Die ESC-Taste erfüllt eine bestimmte Aufgabe: Im allgemeinen zeigt der Benutzer damit an, daß er das Spiel beenden möchte. Unser Programm überprüft, ob die ESC-Taste betätigt wird. Wenn dies der Fall ist, leuchtet das Wort ESC auf dem Bildschirm auf. Damit zeigt das Programm dem Benutzer an, daß die ESC-Taste betätigt wurde. Durch das Drücken der RETURN-Taste wird die Dateneingabe beendet. Das Betätigen jeder anderen Taste macht die Eingabe von ESC wieder rückgängig. Das Programm verbleibt weiter in der Phase der Dateneingabe.

Wie das Programm auf das Betätigen der ESC-Taste reagiert, wird durch die Befehle des Programmierers bestimmt.

Anwendung des „Universelle Eingabe“-Unterprogramms

Dieses Unterprogramm beginnt in Zeile 10000. Die folgenden REM-Anweisungen stehen vor Beginn des Unterprogramms.

```

9990 :
9991 :
9992 REM  ** EINGABE UND WIEDERHOLUNG EINES TEXTES MIT RETURN **
9993 REM  EINTRITT:CURSOR WIRD ZUM BEGINN DES EINGABE-FELDES GESETZT
9994 REM           YW  FELD LAENGE
9995 REM           YF$ PLATZHALTER FUER SPAETERE ZEICHEN
9996 REM  AUSGANG: Z$  TEXT
9997 REM           Z%  -1 (ESC); 0 (NICHT ESC)
9999 :
```

Wie wir sehen, gibt es die Eingangsvariablen YW und YF\$ und die Ausgangsvariablen Z\$ und Z%. Vor dem Eintritt in das Unterprogramm müssen wir den Eingangsvariablen einen Wert zuweisen. Der Eintritt wird durch den Befehl GOSUB 10000 bewirkt. Die YW-Variable bestimmt die Länge des Feldes, d.h. die Anzahl der Zeichen, die unser Programm bei der Eingabe akzeptiert. Wenn der Benutzer ein Wort mit 20 Buchstaben eingeben soll, dann muß im Programm die Anweisung stehen:

200 YW = 20

YF\$ ist ein Platzhalter-Zeichen, die bei vielen Spielen allgemein gebräuchlich sind. Sie erscheinen immer dann, wenn der Benutzer Worte eingeben soll. Die Worte werden dann an die Stelle der Platzhalter gesetzt. Wird YF\$ kein Zeichen zugewiesen, so wird YF\$ als Leerzeichen (blank) angesehen. Auf dem Bildschirm erscheinen deshalb Leerzeichen an der Stelle, die der Benutzer mit Buchstaben ausfüllen soll. Wenn wir aber YF\$ ein Zeichen zuweisen, können wir damit anzeigen, wie viele Buchstaben oder Zahlen einzugeben sind. Hier ist ein Beispiel:

210 YF\$ = "—"

Das folgende Beispiel zeigt, wie man das Unterprogramm in einem Spielprogramm aufruft. Die Feldlänge YW soll 3 betragen, die Platzhalter-Variable ist "X".

```
200 YW = 3: YF$ = "X"
210 PRINT "GIB EINE DREISTELLIGE ZAHL EIN"
220 GOSUB 10000
```

Frage 1: Wie lauten die BASIC-Anweisungen für die Eingangsvariablen eines Wortes mit 10 Buchstaben? Als Platzhalter soll ein Gleichheitszeichen dienen.

Die Ausgangsvariablen erfüllen zwei Funktionen. Z\$ enthält die eingegebenen und auch angenommenen Daten. Dies können sowohl Zahlen als auch Buchstaben sein. Die Variable Z% beträgt - 1, wenn die ESC-Taste gedrückt wird; sie bleibt auf Null, wenn die ESC-Taste nicht betätigt wird. Die ESC-Taste kann für viele Zwecke genutzt werden. Dieses Unterprogramm gibt uns die Freiheit, die Funktionen der ESC-Taste selbst zu wählen. In einigen unserer Programme haben wir folgende Vereinbarung getroffen: Mit dem Drücken der ESC-Taste während des Spiels gibt der Benutzer seinem Wunsch nach Hilfe Ausdruck. Es erscheinen Hinweise und Informationen auf dem Bildschirm. Drückt der Benutzer anschließend nochmals die ESC-Taste, wird das Programm beendet. Will er das Spiel fortsetzen, kann er die RETURN-Taste betätigen. Dies alles wird gesteuert, indem wir die Flexibilität unseres Programms EINGABE EINHEIT ausnutzen.

Die ESC-Funktion kann auch benutzt werden, um zu einem Menü zu gelangen, das mehrere Wahlmöglichkeiten bietet. Zum Beispiel können in einem Kartenspiel die Karten neu gemischt werden, oder ein Spieler kann die Runde verlassen. Es gibt noch eine Menge anderer Beispiele für die Anwendung der ESC-Funktion. In diesem Unterprogramm wird nur festgelegt, ob die ESC-Taste gedrückt wurde. Wie das Programm darauf reagiert, bestimmt der Programmierer selbst.

Hier ist der restliche Teil des „Universelle Eingabe“-Unterprogramms.

```
10000 IF LEN (YF$) < > 1 THEN YF$ = " ": REM INITIALISIERE YF$ WENN
      NOTWENDIG
10010 YH% = PEEK (36) + 1: REM H-POS
10020 YV% = PEEK (37) + 1: REM V-POS
10100 GOSUB 10500: REM SETZE DAS EINGABE FELD ENTSPRECHEND YF$ UND INI
      TIALISIERE
10110 GET Z1$
10120 IF Z1$ = CHR$ (13) THEN RETURN
10130 IF Z1$ < > CHR$ (27) THEN 10200: REM ESC
10140 GOSUB 10500
10150 FLASH : PRINT "ESC"; CHR$ (8);: NORMAL
10160 Z% = - 1: REM ESC FLAG
10170 GOTO 10110
10200 IF Z1$ < > CHR$ (8) THEN 10300: REM LINKER PFEIL
10210 IF Z% = - 1 OR LEN (Z$) < = 1 THEN 10100: REM ABFRAGE ESC ODE
      R EIN ZEICHEN ODER WENIGER
10220 PRINT CHR$ (8);YF$;CHR$ (8);: REM LOESCHE EIN ZEICHEN
```

```

10230 Z$ = LEFT$ (Z$, LEN (Z$) - 1)
10240 GOTO 10110
10300 IF Z1$ < " " THEN 10110: REM   IGNORIERE ANDERE KONTROLLZEICHEN
10310 IF Z% = - 1 THEN GOSUB 10500: REM   LOESCHE ESCAPE BEDINGUNG
10320 IF LEN (Z$) < YW THEN 10400
10330 IF YW = 0 THEN 10110: REM   WIEDERHOLE NICHT WENN LAENGE =0
10340 PRINT CHR$ (8);: REM   BEREITS BEI MAXIMALER LAENGE
10350 IF LEN (Z$) = 1 THEN Z$ = ""
10360 IF LEN (Z$) > 1 THEN Z$ = LEFT$ (Z$, LEN (Z$) - 1)
10400 PRINT Z1$;: REM   WIEDERHOLE UND FUEGE ZEICHEN HINZU
10410 Z$ = Z$ + Z1$
10420 GOTO 10110
10500 HTAB YH%: VTAB YV%: FOR Z = 1 TO YW: PRINT YF$;: NEXT : REM   BES
      CHREIBE FELD ENTSPRECHEND YF$
10510 PRINT " ";: REM   UND LOESCHE MOEGLICHEN CURSOR
10520 IF YW < 2 THEN FOR Z = YW + 1 TO 3: PRINT " ";: NEXT : REM   LOES
      CHE MOEGLICHES ESC WENN FELD NICHT LANG GENUG
10530 HTAB YH%: VTAB YV%
10540 Z$ = ""
10550 Z% = 0
10560 RETURN

```

Wir sehen, daß alle eingegebenen Daten mit einer Textvariable Z\$ belegt werden.

Wir schreiben jetzt einige Anweisungen, die uns die Eingabe eines Wortes mit vier Buchstaben ermöglichen. Das Platzhalter-Zeichen soll ein Punkt sein. Eine Abfrage von ESC (GOTO 4000) ist auch enthalten. Wenn die ESC-Taste nicht betätigt wird, kann der Benutzer ein weiteres Wort mit 10 Buchstaben eingeben.

```

200 YW = 4: YF$ = " . "
210 PRINT "GEBEN SIE IHRE LOESUNG EIN:"
220 GOSUB 10000
230 :
240 IF Z% = - 1 THEN 4000: REM ESC-TEST
260 :
270 YW = 10: YF$ = " . "
280 PRINT "GEBEN SIE EIN WEITERES WORT EIN:"
290 GOSUB 10000
300 REM PROGRAMM WIRD FORTGESETZT
.
.
.
3999 STOP

```

Starten Sie das Programm und probieren Sie einige Variationen aus. Beantworten Sie anschließend folgende Fragen:

Frage 2: Versuchen Sie mehr Buchstaben einzugeben als vom Programm akzeptiert werden. Was geschieht?

Frage 3: Was geschieht, wenn die Taste mit dem nach links gerichteten Pfeil gedrückt wird?

Frage 4: Was bewirkt die Eingabe von CTRL C?

Frage 5: Wie reagiert das Programm auf die ESC-Taste?

Einige technische Bemerkungen

Dieses kleine Unterprogramm ist in seinen Auswirkungen sehr leistungsfähig. Wenn im Programm kein Zeichen angegeben wird, belegt das Unterprogramm in Zeile 10000 die Platzhalter-Variable mit dem vorbestimmten Leerzeichen (blank). Die Zeilen 10010 und 10020 führen zum späteren Gebrauch Variablen für die Cursor-Position ein. Das Unterprogramm in Zeile 10500 druckt die Platzhalter-Variable auf dem Bildschirm ein und zeigt dadurch, daß nun die Eingabe von Daten erwartet wird.

Die einzige Stelle, an der die Daten direkt eingegeben werden, ist die GET-Anweisung in Zeile 10110. Die Dateneingabe erfolgt mit einer Textvariable. Deshalb werden sowohl Zahlen und Buchstaben als auch spezielle Symbole akzeptiert. Die RETURN-Funktion wird in Zeile 10120, die ESC-Funktion in Zeile 10130 abgefragt. Anschließend behandelt das Programm die Löschroutine mit der "←"-Taste (10200 bis 10230), ignoriert alle unerwünschten Zeichen (10300) und überprüft die Länge der eingegebenen Worte und Zahlen (10330 bis 10350).

Frage 6: Welche Funktion hat die Zeile 10410?

Das Unterprogramm „Zahlen Eingabe“

Das vorherige Programm kann alle eingegebenen Daten akzeptieren: Zahlen, Buchstaben und spezielle Symbole. Wir haben zwei Unterprogramme zur Eingabe numerischer Daten entwickelt: ein Unterprogramm zur Eingabe von Integer-Zahlen und eins zur Eingabe von Dezimalzahlen. Hier sind die ersten Programmzeilen des „Zahlen Eingabe“-Unterprogramms. Sie werden sowohl bei Integer-Zahlen als auch bei Dezimalzahlen benutzt.

```
10592 REM      * EINGABE ZAHL *
10593 REM      EINGANG: BEDINGUNGEN FUER EINGABE TEXTSATZ
10594 REM      AUSGANG:  Z% -1 (ESC); 0 (UNGUELTIG); 1 (GANZE ZAHL); 2 (DEZ
      IMALZAHL)
10595 REM      Z WERT (WENN GUELTIG)
10599 :
10600 GOSUB 10000: REM      LESE TEXT
10610 IF Z% = - 1 OR LEN (Z%) = 0 THEN RETURN : REM      NUR ESC ODER RE
      TURN(Z%=0)
10620 Z% = 1: REM      SETZE GUELTIGES ZEICHEN
10630 FOR Z1 = 1 TO LEN (Z%):Z1% = MID% (Z%,Z1,1)
10640 IF Z1% = "." AND Z% = 1 THEN Z% = 2: GOTO 10660: REM      BESTIMMUNG
      DES ERSTEN DEZIMALPUNKTES
10650 IF (Z1% < "0" OR Z1% > "9") AND (Z1% < > "-" AND Z1 > 1) THEN Z% =
      0: REM      UNGUELTIG WENN KEINE ZAHL ODER EIN "-" ZEICHEN
10660 NEXT
10670 Z = VAL (Z%): REM      WERT NUR BEI GUELTIGER FLAG (Z%=1 OR 2)
10680 RETURN
10991 :
```

Die Eingabe von Integer-Zahlen

Manchmal wird vom Benutzer die Eingabe einer Integer-Zahl innerhalb eines bestimmten Intervalls verlangt, z.B. zwischen 1 und 100; für negative Zahlen wird das Minuszeichen (–) verwendet. Wir wollen jetzt ein spezielles Unterprogramm vorstellen. Der Eintritt in dieses Unterprogramm erfolgt in Zeile 11000.

Hier ist das Listing zur Eingabe von Integer-Zahlen:

```
10991 :
10992 REM  ** EINGABE GANZE ZAHL **
10993 REM  EINGANG:BEDINGUNGEN FUER EINGABE TEXT SATZ
10994 REM          YL MINIMALE GANZE ZAHL
10995 REM          YH MAXIMALE GANZE ZAHL
10996 REM  AUSGANG: Z% -1 (ESC); 0 (UNGUELTIGE AZHL); 1 (GUELTIGE ZAHL
)
10997 REM          Z -WERT  (WENN GANZE ZAHL GUELTIG)
10999 :
11000 GOSUB 10600: REM  EINGABE ZAHL
11010 IF Z% < 1 THEN RETURN : REM  ESC ODER UNGUELIG
11020 IF Z% = 2 THEN Z% = 0: RETURN : REM  UNGUELIG WENN DEZIMALPUNKT
11030 IF Z < YL OR Z > YH THEN Z% = 0: REM  UNGUELIG WENN AUSSERHALB D
ER INTERVALLGRENZEN
11040 RETURN
```

Die Eingangsvariablen bestimmen die obere und untere Grenze der Integer-Zahlen, weiterhin bestimmen YW und YF\$ die Länge des Feldes und das Platzhalter-Zeichen. Im folgenden Listing werden die Eingangsvariablen so festgesetzt, daß nur eine dreistellige Zahl zwischen 250 und 750 eingegeben werden kann.

```
200 YW = 3: YF$ = "—"
210 YL = 250: YH = 750
220 PRINT "GIB EINE DREISTELLIGE ZAHL EIN:"
230 GOSUB 11000
240 :
```

Die Ausgangsvariablen dieses Unterprogramms unterscheiden sich von den vorhergehenden. Wird die ESC-Taste gedrückt, dann ist $Z\% = -1$. Wenn die eingegebene Zahl in den Bereich zwischen 250 und 750 fällt, wird $Z\% = 0$ gesetzt. Das bedeutet, daß der Benutzer in seinem eigenen Programm eine Überprüfung vornehmen kann. Der Spieler kann so angewiesen werden, eine Zahl innerhalb des zufälligen Intervalls anzugeben. Die Variable Z enthält die eingegebene und vom Programm akzeptierte Zahl. Die nächsten Programmzeilen demonstrieren, wie die Ausgangsvariablen gesetzt werden können:

```
250 IF Z% = -1 THEN 5000: REM ESC TEST
260 IF Z% = 0 THEN PRINT "BITTE GEBEN SIE EINE ZAHL
    ZWISCHEN"; YL; "UND"; YH; "AN.": GOTO 200: REM TEST
    OB ZAHL UNGUELIG
270 IF Z = N THEN 4000: REM RICHTIGE ANTWORT
280 :
```

Die Eingabe von Dezimalzahlen

Zur Eingabe von Dezimalzahlen benutzen wir das folgende Unterprogramm:

```
11091 :
11092 REM  ** EINGABE DEZIMALZAHL **
11093 REM  EINGANG:BEDINGUNGEN FUER EINGABE TEXTSATZ
11094 REM          YL MINIMALER WERT
11095 REM          YH MAXIMALER WERT
11096 REM  AUSGANG: Z% -1 (ESC); 0 (UNGUELTIG); 1 (GANZE ZAHL); 2 (DEZIMALZ
    AHL)
11097 REM          Z-WERT      (WENN GUELTIG)
11099 :
11100 GOSUB 10600: REM  EINGABE ZAHL
11110 IF Z% < 1 THEN RETURN : REM  ESC ODER UNGUELTIG
11120 IF Z < YL OR Z > YH THEN Z% = 0: REM  UNGUELTIG WENN AUSSERHALB DER IN
    Tervallgrenzen
11130 RETURN
11191 :
```

Der Eintritt erfolgt in Zeile 11100. Die Eingangsvariablen bleiben erhalten: YL und YH für die Intervallgrenzen, YW und YF\$ für Feldlänge und Platzhalter-Symbol. Die Ausgangsvariable Z% kann einen weiteren Wert annehmen. Sie erhält den Wert 2, wenn die eingegebene Zahl einen Dezimalpunkt enthält. Ansonsten ist ihre Funktion die gleiche wie im vorherigen Unterprogramm.

Das „J/N“-Unterprogramm

Eine besondere Form der Dateneingabe liegt vor, wenn nur ein einzelner Buchstabe benötigt wird. Ein typischer Fall ist:

MOECHTEN SIE INFORMATIONEN (J/N):

Das nachfolgende spezielle Unterprogramm akzeptiert nur die beiden Buchstaben J und N:

```
11191 :
11192 REM  ** EINGABE J ODER N **
11193 REM  EINGANG:CURSOR UND YF$ GESETZT
11194 REM  AUSGANG: Z% -1 (ESC); 0 (WEDER J NOCH N ); 1 (J), 2 (N)
11199 :
11200 Y$ = "JN": REM  BENUTZE UNTERPROGRAMM ZUR EINGABE EINES EINZELNEN ZEIC
    HENS
11291 :
11292 REM  ** EINGABE EINES EINZELNEN ZEICHEN UND VERBINDEUNG MIT GUELTIGEM
    TEXT **
11293 REM  EINGANG:CURSOR UND YF$ GESETZT
11294 REM          Y$ TEXT VON VERBINDUNGSZEICHEN
11295 REM  AUSGANG: Z% -1 (ESC); 0 (ZEICHEN NICHT IM TEXTSATZ); J (J'TES ZE
    ICHEN IM VERBINDUNGSTEXT)
11299 :
11300 YW = 1: REM  SETZE FELD-LAENGE
11310 GOSUB 10000
11320 IF Z% = - 1 OR LEN (Z$) = 0 THEN RETURN : REM  NUR ESC ODER RETURN(
    Z%=0)
11330 Z% = 0: REM  SETZE ZEICHEN FUER NICHT VERBUNDEN
11340 FOR Z1 = 1 TO LEN (Y$)
11350 IF Z$ = MID$ (Y$,Z1,1) THEN Z% = Z1: REM  VERBINDE MIT POSITION Z1
11360 NEXT
11370 RETURN
```

Anwendung des „J/N“-Unterprogramms

Das Programm akzeptiert nur J für Ja und N für Nein. Zur Anwendung genügen die folgenden Zeilen:

```
200 YF$ = "—": REM PLATZHALTER-SYMBOL
210 GOSUB 11200
220 :
```

Die Ausgangsvariable Z% erhält den Wert -1, wenn die ESC-Taste benutzt wurde. Sie ist Null, wenn weder J noch N gedrückt wurde. Gibt der Benutzer J ein, erhält sie den Wert 1. Bei Eingabe von N wird Z% = 2 gesetzt. Die Abfrage von Z% nach Verlassen des Unterprogramms kann wie folgt aussehen:

```
230 IF Z% = -1 THEN 5000: REM ESC TEST
240 IF Z% = 0 THEN PRINT: PRINT "BITTE GEBEN SIE NUR J
    ODER N EIN": GOTO 200: REM NICHT-GUELFIGE EINGABE
250 IF Z% = 1 THEN GOSUB 8000: REM DRUCKE INFORMATIONEN WENN J EINGEGEBEN
260 REM FORTSETZUNG DES PROGRAMMS
```

Das Unterprogramm „Einzelne Zeichen Eingabe“

Ein weiteres Unterprogramm von EINGABE EINHEIT gibt uns die Möglichkeit, jeden beliebigen Buchstaben einzugeben, nicht nur J oder N. Das Unterprogramm beginnt in Zeile 11300. Vor dem Aufruf des Unterprogramms muß wieder die Platzhalter-Variable YF\$ spezifiziert werden. Y\$ enthält diesmal alle Zeichen, die akzeptiert werden sollen. Wenn wir zum Beispiel die Eingabe der Buchstaben A, E, I, O und U erlauben, werden die Variablen wie folgt bestimmt:

```
250 YF$ = "—"
260 Y$ = "AEIOU"
270 PRINT "GIB EINEN BUCHSTABEN EIN:";
280 GOSUB 11300
```

Die Ausgangsvariable ist immer noch Z%, doch ihr Zahlenwert hat jetzt eine andere Bedeutung erhalten. Ist Z% = -1, wurde die ESC-Taste gedrückt. Der eingegebene Buchstabe ist ungültig, wenn Z% den Wert Null annimmt. Sonst ist Z% immer eine positive Zahl, die uns die Nummer des Buchstabens in Y\$ angibt. Zum Beispiel hat der Buchstabe I die Nummer 3, da I der dritte Buchstabe in Y\$ ist (AEIOU).

Das Unterprogramm „Pause oder Tastendruck“

Nachdem der Spieler eine Antwort eingegeben hat, erhebt sich die Frage: Wie lange soll das Programm mit der Reaktion warten?

Der Computer könnte zum Beispiel antworten: "DIESER BUCHSTABE IST NICHT GUELFIG" oder "WAEHLE EINE ZAHL ZWISCHEN 1 UND 40" oder irgendetwas ähnliches. Der Spieler benötigt Zeit, um diese Antworten

zu lesen und zu verarbeiten. Erst danach kann das Programm fortfahren. Bei einer kurzen Pause hat besonders der unerfahrene Spieler zu wenig Zeit zur Verfügung. Ist die Pause zu lang, werden die fortgeschrittenen Spieler ungeduldig. Mit dem „Pause oder Tastendruck“-Unterprogramm können wir eine für Anfänger ausreichend lange Pause in unser Programm aufnehmen. Die Pause kann aber durch Betätigen einer Taste sofort beendet werden. Im allgemeinen fragt das Programm anschließend nach einer neuen Eingabe. Die bereits gedrückte Taste wird als Teil der neuen Eingabe angesehen.

Wir empfehlen den Einbau dieser Eingabe-Möglichkeiten in die Spielprogramme. Lassen Sie den Spieler diese Tastenfunktion selbst entdecken. Wir glauben, daß diese Möglichkeit nicht durch zusätzliche Bedienungshinweise erklärt werden muß. Es verwirrt bloß den Anfänger, und der Bildschirm wird unübersichtlich. Dieses Unterprogramm kann aus EINGABE EINHEIT herausgelöst werden und als selbständige Einheit in Spielprogrammen eingesetzt werden.

```

11391 :
11392 REM  ** PAUSE FESTE LAENGE ODER BIS TASTENDRUCK **
11393 REM  EINGANG:YP  LAENGE DER PAUSE IN INTERNEN ZEITEINHEITEN
11394 REM              0  WARTET AUF TASTENDRUCK
11395 REM  AUSGANG:  Z%  -1 (ESC); 0 (PAUSE ABGELAUFEN); 1 (TASTENDRUCK BEVOR
      R PAUSE ABGELAUFEN)
11396 REM              Z  TASTENDRUCK(ASCII-WERT  + 128)
11399 :
11400 POKE  - 16368,0: REM  LOESCHE BISHERIGES ZEICHEN
11410 Z1 = 0: REM  INITIALISIERE ZAEHLER (* EINGANG FUER EINZELNE TASTEN*)
11420 Z1 = Z1 + 1
11430 Z = PEEK ( - 16384)
11440 IF Z > = 128 THEN Z% = 1 - Z * (Z = 155): RETURN : REM  TASTENDRUCK; A
      BFRAGE FUER ESC DANN RETURN
11450 IF Z1 < YP OR YP = 0 THEN 11420
11460 Z% = 0: REM  PAUSE ABGELAUFEN

```

Die Eingangsvariable YP ist gleich Null, falls sie nicht anders festgelegt wird. Wenn YP = 0 gilt, muß der Benutzer eine Taste drücken, um das Spiel fortsetzen zu können. Ansonsten gibt YP die Länge der Pause in rechnerinternen Zeiteinheiten an. Probieren Sie einige Werte für YP aus.

Die Ausgangsvariable Z% hat den Wert -1, wenn die ESC-Taste gedrückt wurde. Sie nimmt den Wert 0 an, wenn die Pause abgelaufen ist, und erhält den Wert 1, wenn der Spieler eine Taste vor Ablauf der Pause betätigt. Hier ist ein Beispiel, das zeigt, wie die Ein- und Ausgangsvariablen im Programm angewendet werden:

```

300 REM INFORMATIONEN
310 :
320 LET YP = 100
330 GOSUB 11400
340 :
350 IF Z% = - 1 THEN 5000: REM ESC TEST
360 IF Z% = 0 THEN PRINT "WENN SIE MEHR ZEIT BENOETIGEN,
      FRAGEN SIE IHREN LEHRER!": GOTO 300
370 REM FORTSETZUNG DES PROGRAMMS

```

Das Unterprogramm „Tastendruck ohne Wiedergabe“

Ein weiteres, allgemeines Problem ergibt sich aus der Frage: Wie lange soll man nach dem Ausdruck von Informationen oder anderen Mitteilungen warten? Die Lesefähigkeiten der Spieler oder der Grad ihrer Vertrautheit mit dem Spiel sind oft recht unterschiedlich. Deshalb ist es wünschenswert, daß der Spieler dem Programm selbst mitteilt, wann es weitergehen soll, und zwar durch Tastendruck. Das ist eine bessere Lösung, als den Programmierer die Länge der Pausen im voraus abschätzen zu lassen.

Es gibt zwei Versionen dieses Unterprogramms. Die erste heißt: „Tastendruck ohne Wiedergabe, ohne vorheriges Zeichen“. Sie beginnt in Zeile 11500. Das zweite Unterprogramm „Tastendruck ohne Wiedergabe, mit vorherigem Zeichen“ beginnt in Zeile 11600. Beide Programme warten auf die Eingabe, bis eine Taste betätigt wird (das zusätzliche RETURN ist nicht erforderlich). Das zugehörige Zeichen wird aber nicht auf dem Bildschirm wiedergegeben. Im Gegensatz zum „Pause oder Tastendruck“-Programm wird das eingegebene Zeichen anschließend sofort gelöscht. Das Zeichen ist also nicht ein Teil der folgenden Dateneingabe. Die Ausgangsvariable Z% beträgt wieder -1, wenn die ESC-Taste betätigt wurde. Nach Gebrauch einer anderen Taste ist $Z\% = 1$. Die Variable Z erhält den ASCII-Wert der gedrückten Taste zuzüglich 128. Eingangsvariablen sind keine vorhanden.

Der Unterschied zwischen den beiden Unterprogrammen besteht in der Möglichkeit eines vorangehenden Tastendrucks. Die APPLE-Hardware besitzt eine „Ein-Zeichen-Memory“, die jeweils einen Zustand entsprechend dem Wert der zuletzt gedrückten Taste einnimmt. Das Einlesen von der Tastatur bedingt das Warten auf die Festlegung dieses Zustandes, das Ablesen des eingegebenen Zeichens und schließlich das Löschen des vorherigen Speicherzustandes. Damit wird angezeigt, daß keine weitere Taste gedrückt wurde. (Im APPLE II-Handbuch finden sich nähere Einzelheiten.) Das Unterprogramm „ohne vorheriges Zeichen“ ab Zeile 11500 löscht zuerst den bisherigen Speicherinhalt und wartet dann auf den nächsten Tastendruck. Durch das Löschen des Speicherinhalts ist der Benutzer nun gezwungen, nach dem Ausdruck von Informationen (oder anderer Dinge) eine Taste zu betätigen. Anschließend wartet das Programm noch einmal auf einen Tastendruck. Das Unterprogramm „mit vorherigem Zeichen“ gibt dem Benutzer die Möglichkeit, die Pause vorwegzunehmen bzw. sie zu vermeiden. Die Pause wird allerdings auch dann übersprungen, wenn der Benutzer aus Unaufmerksamkeit eine zusätzliche Taste betätigt. Für die Anwendungen dieses Kapitels empfehlen wir die Benutzung des Unterprogramms „mit vorangehendem Zeichen“.

Für interessierte Leser weisen wir darauf hin, daß eine andere Version dieses Unterprogramms in dem Spiel SIMON in Kapitel 7 benutzt wird. In Zeile 3000 des SIMON-Listings wird die nächste Musikenote abgefragt. Das THEN-Kommando entscheidet anschließend, an welcher Stelle die Note wiedergegeben wird.

In den Programmen aus Kapitel 1 bis 4 haben wir keine der hier beschriebenen Techniken zur Dateneingabe benutzt. Wenn der Leser einige der früheren Programme verwenden möchte, müssen diese mit dem Programm EINGABE EINHEIT verbunden und die notwendigen Anpassungen in Bezug auf die Ein- und Ausgangsvariablen vorgenommen werden.

Zusammenfassung der Unterprogramme zur Dateneingabe

Zur Benutzung der Unterprogramme ist es am einfachsten, EINGABE EINHEIT vollständig mit den eigenen Programmen zu verbinden. Wird nur das „Pause oder Tastendruck“-Unterprogramm benötigt, kann der Rest von EINGABE EINHEIT gelöscht werden. Das vollständige Unterprogramm EINGABE EINHEIT benötigt jedoch nicht viel Speicherplatz. Es lohnt sich kaum, dieses in Teile zu zerlegen und nur die benötigten Teile mit dem eigenen Programm zu verbinden. Verwenden Sie das ganze Programm. Das ist viel einfacher!

Unterprogramm „Universelle Eingabe“

Aufruf: GOSUB 10000

Eingangsvariablen: YW: Feldlänge
YF\$: Platzhalter-Symbol (wenn kein Wert bestimmt wird, ist YF\$ das Leerzeichen)

Ausgangsvariablen: Z\$: Eingegebener Text
Z%: - 1 (ESC); 0 (nicht ESC)

Unterprogramm „Zahlen Eingabe“

Dieses Unterprogramm benötigt das „Universelle Eingabe“-Programm.

Integer Zahlen:

Aufruf: GOSUB 11000

Eingangsvariablen: YW: Feldlänge
YF\$: Platzhalter-Symbol
YL: Minimaler Wert
YH: Maximaler Wert

Ausgangsvariablen: Z: Zahlenwert
Z%: - 1 (ESC); 0 (ungültige Integer-Zahl); 1 (gültige Integer-Zahl)

Dezimalzahlen:

Aufruf: GOSUB 11100

Eingangsvariablen: YW: Feldlänge
YF\$: Platzhalter-Symbol
YL: Minimaler Wert
YH: Maximaler Wert

Ausgangsvariablen: Z: Zahlenwert
Z%: -1 (ESC); 0 (ungültige Zahl); 1 (gültige Integer-Zahl); 2 (Dezimalzahl)

Unterprogramm „J/N“

Dieses Unterprogramm benötigt das „Universelle Eingabe“-Programm.

Aufruf: GOSUB 11200

Eingangsvariable: YF\$: Platzhalter-Symbol

Ausgangsvariable: Z%: -1 (ESC); 0 (weder J noch N); 1 (J); 2 (N)

Unterprogramm „Einzelne Zeichen Eingabe“

Dieses Unterprogramm benötigt das „Universelle Eingabe“-Programm.

Aufruf: GOSUB 11300

Eingangsvariablen: Y\$: enthält zulässige Zeichen

YF\$: Platzhalter-Symbol

Ausgangsvariable: Z% -1 (ESC); 0 (Zeichen ist nicht in YF\$ enthalten);
J (Zeichen steht in der j'ten Position in YF\$)

Unterprogramm „Pause oder Tastendruck“

Aufruf: GOSUB 11400

Eingangsvariable: YP: Länge der Pause

Ausgangsvariable: Z%: -1 (ESC); 0 (Pause abgelaufen); 1 (Tastendruck
bevor Pause abgelaufen)

Unterprogramm „Tastendruck ohne Wiedergabe, ohne vorheriges Zeichen“

Aufruf: GOSUB 11500

Eingangsvariable: Keine

Ausgangsvariablen: Z%: -1 (ESC); 1 (anderer Weg)

Z: ASCII-Wert des Tastendrucks + 128

Unterprogramm „Tastendruck ohne Wiedergabe, mit vorherigem Zeichen“

Aufruf: GOSUB 11600

Eingangsvariablen: Keine

Ausgangsvariablen: Z%: -1 (ESC); 1 (anderer Weg)

Z: ASCII-Wert des Tastendrucks + 128

Zusammenfassung von Kapitel 5

In diesem Kapitel haben wir bereits die dritte Programmeinheit vorgestellt, die der Leser zum Schreiben von eigenen Spielprogrammen verwenden kann. Aber auch für andere Programme ist dieses Unterprogramm EINGABE EINHEIT hervorragend geeignet. Es erlaubt die vollständige Kontrolle über alle Daten. Wir können festlegen, welches erlaubte und gültige Dateneingaben sind. In den Spielprogrammen der nun folgenden Kapitel werden wir zeigen, wie wir die Unterprogramme zur Dateneingabe angewendet haben.

Lösungen zu Kapitel 5

- Frage 1: 200 YW = 10: YF\$ = "="
 210 PRINT "GIB EIN WORT MIT 10 BUCHSTABEN EIN:";
 220 GOSUB 10000
- Frage 2: Das letzte Zeichen wird gelöscht und durch den zuletzt getippten Buchstaben ersetzt. Es können nicht mehr Zeichen als erlaubt eingegeben werden.
- Frage 3: Die letzten Zeichen werden gelöscht und können durch neue Zeichen überschrieben werden. Damit kann der Benutzer falsch eingegebene Daten korrigieren.
- Frage 4: Es geschieht nichts. Die CTRL-Taste wird ignoriert.
- Frage 5: Die ESC-Schrift leuchtet solange auf dem Bildschirm auf, bis eine andere Taste betätigt wird. Wenn Daten eingegeben werden sollen, wird die ESC-Taste gelöscht. Die eingegebenen Daten werden akzeptiert.
- Frage 6: Der Ausgangstext Z\$ wird erstellt. Ein Zeichen nach dem anderen wird mit Z\$ verknüpft.

Kapitel 6

Wortspiele

Die Behandlung von Textvariablen, oder das Arbeiten mit Texten, die vom Benutzer beim Spiel eingegeben werden, ist das Grundgerüst einiger „klassischer“ und höchst interessanter Computerspiele. Obwohl durch die technischen Neuerungen Möglichkeiten für Farbe, Grafik und Klang geschaffen wurden, bleiben die Wortspiele weiterhin faszinierend, zum Spielen wie auch zum Programmieren.

Dieses Kapitel behandelt Wortspiele, die von den Textbearbeitungsmöglichkeiten des APPLE Gebrauch machen, und erläutert, wie ein Spiel einer bestimmten Spielerrunde angepaßt wird. Wir werden drei Typen von Wortspielen behandeln: Konstruieren von Geschichten, Wörter raten und Wörter verbinden. Für jeden Typ werden vollständige Spiele entwickelt. Anschließend werden die Funktionen einzelner Programmteile erläutert.

Story

STORY stellt eine Reihe von Fragen und fügt die Antworten in ein vorher festgelegtes Format. Die Struktur des Spiels läßt viele Möglichkeiten offen und kann je nach Teilnehmerkreis modifiziert werden. Das Programm erzeugt aus gegebenen Antworten lustige Satzkombinationen nach Art einiger bekannter und beliebter Kinderspiele.

```
10 REM ...STORY-EINGABE BILDER
11 :
997 :
998 REM ** INITIALISIERUNG **
999 :
1000 DIM RP$(10,10): REM MAXIMALE ANZAHL DER ZUFÄLLEIGEN GRUPPEN * MAXIMALE
    ANZAHL DER FRAGEN IN JEDER GRUPPE
1010 DIM NR(10): REM AKTUELLE ANZAHL DER FRAGEN IN JEDER GRUPPE
1020 DIM A$(10): REM MAXIMALE ZAHL DER ANTWORTEN
1197 :
1198 REM * TITELBILD *
1199 :
1200 GR : HOME : COLOR= 15: GOSUB 15500: REM LOESCHE MIT WEISS
```

```

1210 COLOR= 6: FOR Z = 11 TO 33 STEP 11: REM   BLAUE LINIEN
1220 HLIN 0,39 AT Z - 8
1230 HLIN 0,39 AT Z
1240 NEXT
1250 X$ = "SCHREIB":XV = 4:XC(1) = 0: GOSUB 15300
1260 X$ = "EINE":XV = XV + 11: GOSUB 15300
1270 X$ = "STORY":XV = XV + 11: GOSUB 15300
1280 VTAB 23: HTAB 7: PRINT "MIT RETURN GEHT ES WEITER..";
1290 GOSUB 11500: REM   WARTET AUF TASTENDRUCK
1300 IF Z% = - 1 THEN END : REM   ESC
1997 :
1998 REM   ** INITIALISIERUNG FÜR NÄCHSTE STORY
1999 :
2000 TEXT : HOME
2010 Z = 51000: GOSUB 19000: REM   SETZE DATA-ZEIGER AUF ZUFALLSGRUPPEN
2097 :
2098 REM   * LADE ZUFALLSTEILE VON STORY *
2099 :
2100 RP = 0
2110 J = 0
2120 READ Z$: REM   * SETZE RICHTIGE SATZFOLGE MIT NEUER GRUPPE
2200 RP = RP + 1: REM   NÄCHSTE GRUPPE VON SATZTEILEN
2210 J = 0: REM   ZAHL DER SATZTEILE IN AKTUELLER GRUPPE
2230 IF Z$ = "END" THEN 2300
2240 J = J + 1
2250 RP$(RP,J) = Z$: REM   SPEICHERE SATZTEIL
2260 READ Z$
2270 GOTO 2230
2300 NR(RP) = J: REM   ZAHL DER ZUFALLSTEILE IN GRUPPE RP
2310 READ Z$: REM   ÜBERPRÜFE ZWEITES "END"
2320 IF Z$ = "END" THEN 2500: REM   KEINE WEITEREN ZUFALLSTEILE
2330 GOTO 2200: REM   RICHTIGE SATZKONSTRUKTION,BEGINNE MIT NEUER GRUPPE
2497 :
2498 REM   * STELLE FRAGEN UND SPEICHERE DIE ANTWORT *
2499 :
2500 NQ = 0
2510 HTAB 14: PRINT "*** STORY ***"
2520 VTAB 22: HTAB 11: PRINT "DRÜCKE ";: INVERSE : PRINT "ESC";: NORMAL : PRINT
" FÜR PROGRAMMIERENDE.";
2600 READ Q$: REM   ÜBERPRÜFE OB WEITERE FRAGEN
2610 IF Q$ = "END" THEN 3000: REM   KEINE WEITEREN FRAGEN
2620 NQ = NQ + 1: REM   WEITERE FRAGEN
2630 VTAB 2 * NQ + 4: HTAB 1: PRINT Q$;" ";
2640 YW = 38 - LEN (Q$): REM   MAXIMALE LÄNGE DER ANTWORT
2650 GOSUB 10000
2660 IF Z% = - 1 THEN 6000: REM   ESC
2670 IF LEN (Z$) = 0 THEN 2650: REM   SCHLEIFE FÜR KEINE ANTWORT
2680 A$(NQ) = Z$: REM   SPEICHERE NÄCHSTE ANTWORT
2690 GOTO 2600
2997 :
2998 REM   ** SCHREIBEN VON STORY **
2999 :
3000 HOME : REM   PAUSE VOR DEM SCHREIBEN DER GESCHICHTE
3010 SPEED= 10
3020 VTAB 11: HTAB 9
3030 PRINT "HIER IST DEINE GESCHICHTE.": CHR$ (7);".": CHR$ (7);".": CHR$ (7)

3040 SPEED= 255
3050 YP = 50: GOSUB 11400: REM   PAUSE ODER TASTENDRUCK
3091 :

```

```

3092 REM * SCHREIBE GESCHICHTE AUS DEN SATZTEILEN DER DATA-ANWEISUNGEN *
3093 REM SCHLEIFE FUER WOERTER,DIE AM ENDE DER ZEILE ABGESCHNITTEN WERDEN
3099 :
3100 L = 3: REM LINKER RAND (FUER TEXTFENSTER)
3110 W = 34: REM BREITE
3120 T = 4: REM OBERSTE ZEILE
3130 B = 20: REM UNTERSTE ZEILE
3140 POKE 32,L: REM SETZE TEXTFENSTER
3150 REM SETZEN DER BREITE IST UNNOETIG
3160 POKE 34,T
3170 POKE 35,B
3180 HOME : REM BEWEGE CURSOR ZUR UNTEREN LINKEN ECKE DES FENSTERS
3200 S$ = "": REM INITIALISIERE BILDSCHIRM-ZEILE
3210 READ Z$: REM NAECHSTES STORY ELEMENT
3220 IF Z$ = "END" THEN 3900: REM ENDE VON STORY
3230 IF LEFT$ (Z$,1) = "#" THEN 3500: REM BENUTZE SPEZIFIZIERTE ANTWORT NUMMER
3240 IF LEFT$ (Z$,1) = "$" THEN 3600: REM BENUTZE ZUFALLSTEIL DER ANGEGEBENEN GRUPPE
3300 S$ = S$ + Z$: REM VERBINDE STORY TEILE
3310 IF LEN (S$) < = W THEN 3210: REM BILDSCHIRMZEILE NOCH NICHT GANZ VOL
L
3320 Z = W + 1: REM VERSUCHE DIE ZEILE NACH DEM AM WEITESTEN RECHTS STEHENDEN LEERZEICHEN ABZUBRECHEN
3330 REM STARTE MIT DEM ERSTEN BUCHSTABEN UNTERHALB DER MAXIMALEN LAENGE
3340 IF Z = 1 THEN Z = W + 1: GOTO 3400: REM KEINE LEERZEICHEN VORHANDEN; BENUTZE MAXIMALE LAENGE
3350 IF MID$ (S$,Z,1) = " " THEN 3400: REM LEERZEICHEN IN POSITION Z GEFUNDEN
3360 Z = Z - 1
3370 GOTO 3340
3400 PRINT LEFT$ (S$,Z - 1): REM UNTERBRECHE DIE ZEILE BEIM Z'TEN ZEICHEN
3410 IF Z = LEN (S$) THEN 3200: REM ES BLEIBT NICHTS UEBRIG
3420 S$ = RIGHT$ (S$, LEN (S$) - Z): REM REST DER ZEILE
3430 GOTO 3310: REM UEBERPRUEFE OB NOCH ZU LANG
3500 Z$ = A$( VAL ( RIGHT$ (Z$, LEN (Z$) - 1))) : REM BENUTZE SPEZIFIZIERTE ANTWORT NUMMER
3510 GOTO 3300
3600 Z = VAL ( RIGHT$ (Z$, LEN (Z$) - 1)): REM ZUFALLSGRUPPE BESTIMMT
3610 Z$ = RP$(Z,1 + INT (NR(Z) * RND (1))): REM WAEHLE EIN ZUFALLSELEMENT
3620 GOTO 3300
3900 IF LEN (S$) > 0 THEN PRINT S$: REM * ENDE VON STORY - DRUCKE RESTLICHEN TEIL *
3910 TEXT : REM VOLLES TEXTFENSTER
5997 :
5998 REM ** NOCH EINMAL? **
5999 :
6000 HTAB 1: VTAB 24
6010 PRINT "EINE WEITERE GESCHICHTE(J ODER N)? ";
6020 GOSUB 11200: REM J/N
6030 ON Z% + 2 GOTO 6100,6000,2000,6100: REM ESC, UNGUELTIG, J, N
6100 PRINT : PRINT
6110 PRINT "DANKE FUER'S MITSPIELEN.";
6120 END
50991 :
50992 REM * ZUFAELLIGE SATZTEILE,MIT "END" ABGESCHLOSSEN *
50993 REM JEDE GRUPPE ENDET MIT "END"
50994 REM ZUFALLSGRUPPEN '51000 DATA "END","END"
50999 :
51000 DATA "AM KARNEVAL" , "IN EINER DUNKLEN NACHT", "END": REM * GRUPPE 1 *

```

```

51010 DATA "RANNTEN", "SASSEN", "POKERTEN", "END": REM * GRUPPE 2 *
51020 DATA "AM STRAND", "IN DEN BERGEN", "IN EINEM JAGDHAUS", "IN
      DER SCHULE", "IN DER WUESTE", "END": REM * GRUPPE 3 *
51030 DATA "HOERTEN", "BEMERKTEN", "END": REM * GRUPPE 4 *
51040 DATA "SCHLEIMIGE", "GROSSE", "STRUPPIGE", "WILDE", "END": REM * GRU
      PPE 5 *
51050 DATA "FLIRTETEN", "SPIELTEN SCHACH", "ERZAEHLTEN SICH GESCHICHTEN", "
      POKERTEN", "STREITETEN", "END": REM * GRUPPE 6 *
51060 DATA "BRACHTE IHNEN ALLEN EINEN GEBURTSTAGS-KUCHEN", "FLOG MIT IHNEN
      IN EINEM HUBSCHRAUBER VON DANNEN", "SANG IHNEN EIN SCHLAFLIED", "END": REM
      * GRUPPE 7 *
51070 DATA "END": REM ENDE DER ZUFALLSGRUPPEN
51991 :
51992 REM * FRAGENTEIL, WIRD MIT "END" ABGESCHLOSSEN *
51999 :
52000 DATA "WIE IST DEIN NAME?"
52010 DATA "WAS IST DEINE LIEBLINGSFARBE?"
52020 DATA "WEN MAGST DU BESONDERS GERN?"
52030 DATA "WELCHE TIERE FUERCHTEST DU?"
52040 DATA "WER IST DEIN BESTER FREUND?"
52050 DATA "END"
52991 :
52992 REM * STORY, ENDET MIT "END" *
52993 REM ART DER DATEN:
52994 REM "#NUMMER" = DRUCKE ANTWORT NUMMER
52995 REM "$NUMMER" = DRUCKE EINEN SATZTEIL AUS GRUPPE NUMMER$
52996 REM "END" = ENDE VON STORY (WIRD NICHT GEDRUCKT)
52997 REM SONSTIGES = DRUCKE ALS TEXTVARIABLE
52999 :
53000 DATA "$1", " ", "$2", " ",
53010 DATA "#1", " UND ", "#5"
53020 DATA " ", "$3", ". PLOETZLICH "
53030 DATA "$4", " SIE VIELE ", "$5", " "
53040 DATA "#2", "E ", "#4"
53050 DATA ". DIE "
53060 DATA "#2", "EN ", "#4"
53070 DATA " ABER ", "$6", " MIT IHNEN, DOCH DANN KAM "
53080 DATA "#3"
53090 DATA " UND ", "$7", ". "
53100 DATA "END"
60000 :
60010 REM * COPYRIGHT 1981 BY HOWARD FRANKLIN, PALO ALTO, CA *
60020 :

```

Verbinden Sie dieses Programm mit BILD EINHEIT. Speichern Sie es mit dem Befehl SAVE STORY und tippen Sie RUN.

Dem Programm liegt das original STORY-Programm zugrunde. Es wurde neu überarbeitet, um Eingabe-Überprüfungen und Bildschirm-Formatierung mit aufzunehmen. STORY wurde im Community Computer Center geschrieben und erschien 1976 erstmals in einer amerikanischen Computer-Zeitschrift. STORY wurde als lustiges Spiel entworfen, soll jedoch ebenso eine nützliche Sprachübung für Leseanfänger sein. Die Fragen in diesem Spiel sind von persönlicher Art. Kinder merken sich sehr leicht ihre Antworten. Auf diese Weise ist es nicht schwierig für sie, die Geschichte, die der Computer ausgibt, zu lesen. Das Spiel STORY fragt nach Antworten auf bestimmte Fragen,

fast wie in einem kleinen Gespräch, und unterscheidet sich so von anderen Spielen dieser Art.

Auch gibt es bei STORY weder Gewinner noch Verlierer, im Gegensatz zu den mehr traditionellen Spielen. Daher empfinden größere Kinder und sogar Erwachsene Freude an diesem Spiel. Es kann als eine spielerische Hinführung zu Computern angesehen werden, die viel Spaß bereitet. In der Tat kann STORY von erfahrenen Spielern auch leicht zu einer ‚Einführung in das Programmieren‘ abgewandelt werden. Nach mehrmaligem Benutzen des Programms kann man ein bestimmtes Muster in der Konstruktion der Geschichten erkennen und beginnt so, die Arbeitsweise des Programms zu verstehen. Basierend auf den Erfahrungen mit STORY, kann den Benutzern dieses Spiels die Funktionsweise von Computerprogrammen erläutert werden. Die Fragen sind so gestellt, daß sowohl ein einzelnes Wort als auch längere Formulierungen als Antworten möglich sind. Entsprechen die Antworten in grammatikalischer Hinsicht den gestellten Fragen, so fügen sie sich sauber in den anschließenden Aufbau der Geschichte ein. Die Länge der Antworten ist jedoch begrenzt, wie man aus Programmzeile 2640 ersehen kann.

Frage 1: Was wird geschehen, wenn Sie eine Frage so ändern, daß diese aus 45 Zeichen besteht?

Frage 2: Wie würden Sie das Programm ändern, um andere Fragen zu stellen?

Das zufällige Abbrechen eines Wortes am Ende einer Zeile, auch das kontinuierliche Weiterschreiben am Beginn der nächsten Zeile, wollen wir an dieser Stelle einmal mit „Überschreiben“ benennen. Dieses Überschreiben kann sehr ärgerlich sein. Bedauerlicherweise leiden noch immer viele mit Texten arbeitende Programme daran. Ein Programmteil (Prozedur) von STORY verhindert dieses Überschreiben am Bildschirm. Die Prozedur überprüft den freien Raum (am Ende des Wortes) und unterbricht die Zeile an geeigneter Stelle. Vielleicht möchten Sie diese Prozedur in anderen, von Ihnen geschriebenen Programmen benutzen?

Frage 3: Welcher Teil behandelt das Problem des Überschreibens am Bildschirm?

STORY sollte stets dem speziellen Benutzerkreis angepaßt werden. Formulierungen, Inhalt und Länge der Geschichte müssen den Lesefertigkeiten und der Erfahrung bzw. dem Wissen der Mitspieler entsprechen.

Wir zeigen nun, wie Teile des Textes einer Geschichte geändert werden können. Dazu sehen wir uns die Programmzeilen 50000 bis 51070 an. Sie enthalten zufällig ausgewählte Ausdrücke und Redewendungen für die Geschichte. Jede Folge solcher Satzteile endet mit dem Wort „END“, gefolgt von einer REM-Anweisung. (Das Wort „END“ ist unbedingt erforderlich, die REM-Anweisung kann natürlich auch weggelassen werden.) Beispiel:

```
51000 DATA "AM KARNEVAL", "IN EINER DUNKLEN NACHT",  
"END": REM *RANDOM1*
```


Zeile 51000 zeigt also, daß in RANDOM1 nur zwei Möglichkeiten für eine zufällig ausgewählte Redewendung angeboten werden. Unser Programm besitzt sieben solcher Zusammenstellungen von Satzteilen.

Frage 4: Wie viele Auswahlmöglichkeiten gibt es bei der dritten Zusammenstellung?

Man kann den Text einer Geschichte ändern, indem man die Ausdrücke in diesen Zusammenstellungen ändert. Hierbei muß beachtet werden, daß dieselbe Struktur wie im Programm benutzt wird. Den Aufbau einer Geschichte zu ändern, erweist sich als etwas schwieriger. Eine Geschichte wird nach folgendem Schema aufgebaut:

```
52992 REM * STORY, ENDET MIT "END" *
52993 REM ART DER DATEN:
52994 REM "#NUMMER" = DRUCKE ANTWORT NUMMER
52995 REM "$NUMMER" = DRUCKE EINEN SATZTEIL AUS GRUPPE NUMMER$
52996 REM "END" = ENDE VON STORY(WIRD NICHT GEDRUCKT)
52997 REM SONSTIGES = DRUCKE ALS TEXTVARIABLE
52999 :
53000 DATA "§1"," ","§2"," ",
53010 DATA "#1"," UND ","#5"
53020 DATA " ","§3",". PLOETZLICH "
53030 DATA "§4"," SIE VIELE ","§5"," "
53040 DATA "#2","E ","#4"
53050 DATA ". DIE "
53060 DATA "#2","EN ","#4"
53070 DATA " ABER " , "§6"," MIT IHNEN,DOCH DANN KAM "
53080 DATA "#3"
53090 DATA " UND ","§7",". "
53100 DATA "END"
```

Die Zeilen 52994 bis 52997 zeigen, wie eine solche Geschichte erstellt wird. Es folgt der formale Aufbau der zu spielenden Geschichte. Um den Ausdruck der vom Benutzer gegebenen Antwort auf dem Bildschirm erscheinen zu lassen, benutzen wir das #-Symbol, gefolgt von der Frage-Antwort-Nummer, wie in Zeile 53080. Die dritte Antwort des Benutzers wird entsprechend der DATA-Anweisung in Zeile 53080 ausgedruckt. Um einen der zufälligen Satzteile auszusuchen, benutzen wir das \$-Symbol, gefolgt von der Nummer einer dieser RANDOM-Gruppen. So bewirkt Zeile 53000 die Auswahl eines der Satzteile aus der mit GRUPPE1 gekennzeichneten Gruppe in Zeile 51000. Alle anderen Texte in der Geschichte werden so ausgedruckt, wie sie eingegeben wurden (zum Beispiel die Wörter in den Zeilen 53020 und 53070). Die Geschichte wird beendet durch die Eingabe von "END", wie in Zeile 53100 zu sehen ist.

Sie werden feststellen, daß das geschickte Zusammenfügen der Satzteile und Antworten der Mitspieler sowohl Praxis als auch Erfahrung erfordert. Je mehr Sie spielen, desto besser werden die Geschichten werden. Ihre Freunde, gleich welchen Alters, werden Ihre Geschichten genießen.

Blockout

BLOCKOUT ist ein Wortratespiel und hat seinen Ursprung in einem amerikanischen Kinderspiel namens Hangman. Der Spieler hat eine feste Anzahl von Versuchen, um ein unbekanntes Wort zu erraten. Wenn er das Wort nicht errät, wird er am Galgen aufgehängt. BLOCKOUT basiert auf dem gleichen Prinzip, doch haben wir die Grausamkeit, die bei Hangman im Hintergrund besteht, weggelassen. Auch kann die festgelegte Anzahl von Versuchen zum Erraten des Wortes verändert werden. BLOCKOUT präsentiert das beliebte Kinderspiel in neuer Form, indem es sich die Grafikfähigkeiten des APPLE zunutze macht.

```
10 REM ...BLOCKOUT - EINGABE KLANG BILD
11 :
997 :
998 REM ** INITIALISIERUNG **
999 :
1000 DIM WD$(50)
1010 ML = 16: REM MAXIMALE LAENGE DES UNBEKANNTEN WORTES
1020 HG = 8: REM HORIZONTALE POSITION FUER RATEVERSUCHE
1030 HC = HG + ML + 1: REM HORIZONTALE POSITION FUER ANHALTSPUNKTE
1040 VG = 21: REM VERTIKALE POSITION FUER RATEVERSUCHE
1050 VC = VG: REM VERTIKALE POSITION FUER ANHALTSPUNKTE
1100 NW = 0: REM * ZAEHLE DIE WOERTER AUND SPEICHERE SIE IN WD$(J) *
1110 Z = 51000: GOSUB 19000: REM LESE DATEN AUS ZEILE 51000
1120 READ Z$
1130 IF Z$ = "END" THEN 1200: REM KEINE WEITEREN WORTE
1140 IF LEN (Z$) > ML THEN 1120: REM LOESCHE ZU LANGE WOERTER
1150 NW = NW + 1: REM NAECHSTES WORT
1160 WD$(NW) = Z$
1170 GOTO 1120
1200 NG = 10: REM VORBESTIMMTE ANZAHL DER FALSCHEN VERSUCHE
1210 GU = 15: REM OBERE SCHRANKE FUER RATEVERSUCHE
1220 GL = 5: REM UNTERE SCHRANKE
1300 DIM BK%(15,1): REM H/V POSITIONEN DER BLOECKE
1310 BH = 2: REM HORIZONTALE LAENGE
1320 BV = 3: REM VERTIKALE HOEHE
1397 :
1398 REM * TITELBILD *
1399 :
1400 GR : HOME : COLOR= 1: GOSUB 15500: REM HINTERGRUND IN ROT
1410 COLOR= 0: FOR J = 1 TO 80: REM ZUFALLSBLOECKE IN SCHWARZ
1420 XH = INT (37 * RND (1)) + 1: XV = INT (36 * RND (1)) + 1: REM ZUFUEL
LIGE POS
1430 FOR Z = 0 TO 2: HLIN XH,XH + 1 AT XV + Z: NEXT
1440 NEXT
1450 XC(1) = 15: XV = 12: X$ = "BLOCK": GOSUB 15300
1460 XV = XV + XB + 2: X$ = "OUT": GOSUB 15300
1470 VTAB 23: HTAB 6: PRINT "MIT RETURN GEHT ES WEITER..";
1480 GOSUB 11500: REM WARTE AUF TASTENDRUCK
1490 IF Z% = - 1 THEN END : REM ESC
1900 GOSUB 9000: REM INFORMATIONEN
1997 :
1998 REM ** INITIALISIERUNG FUER EIN NEUES SPIEL **
1999 :
2000 IF NW = 0 THEN 6800: REM KEINE WEITEREN WORTE
2010 Z = INT (NW * RND (1)) + 1
```

```

2020 SW$ = WD$(Z): REM NIMM EINES DER BISHER NICHT BENUTZTEN WOERTER
2030 WD$(Z) = WD$(NW): REM ERSETZE DAS GEWAELHTE WORT DURCH DAS NAECHSTE UN
      BENUTZTE WORT
2040 NW = NW - 1: REM UND VERRINGERE DIE LISTE DER UNBENUTZTEN WOERTER UM 1
2100 L$ = "": REM INITIALISIERE L$,U$,C$, UND BL$
2110 U$ = ""
2120 C$ = ""
2130 FOR J = 1 TO 26
2140 L$ = L$ + CHR$(64 + J): REM NAECHSTER BUCHSTABE DES ALPHABETS
2150 U$ = U$ + " ": REM EIN WEITERES LEERZEICHEN
2160 C$ = C$ + "-": REM EIN WEITERES -
2170 NEXT
2180 C$ = LEFT$(C$, LEN(SW$)): REM KUERZE DIE HINWEISE AUF DIE LAENGE DES
      UNBEKANNTEN WORTES
2190 BL$ = U$ + U$: REM 52 LEERZEICHEN
2300 GOSUB 9500: REM SETZE DIE BLOECKE
2997 :
2998 REM ** NAECHSTER RATEVERSUCH **
2999 :
3000 VTAB VG: HTAB HG
3010 YW = 1: GOSUB 10000
3020 IF Z% = - 1 THEN 6200: REM ESC
3030 IF Z$ < "A" OR Z$ > "Z" THEN 4700: REM KEIN BUCHSTABE DES ALPHABETS
3100 REM UEBERPRUEFE OB DER BUCHSTABE SCHON GERATEN WURDE
3110 Z = ASC(Z$) - ASC("A") + 1: REM POSITION IN U$
3120 IF MID$(U$,Z,1) < > " " THEN 4600: REM BEREITS BENUTZT
3130 REM ERNEUERE L$ UND U$ UND DRUCKE AUS
3140 REM SPEZIELLER FALL DER TEXT FUNKTIONEN BEI ERSTEM ODER LETZTEN BUCHST
      ABEN
3150 IF Z = 1 THEN L$ = " " + RIGHT$(L$,25):U$ = Z$ + RIGHT$(U$,25)
3160 IF Z = 26 THEN L$ = LEFT$(L$,25) + " ":U$ = LEFT$(U$,25) + Z$
3170 IF Z > 1 AND Z < 26 THEN L$ = LEFT$(L$,Z - 1) + " " + RIGHT$(L$,26 -
      Z):U$ = LEFT$(U$,Z - 1) + Z$ + RIGHT$(U$,26 - Z)
3180 VTAB 23: HTAB HG + 2: PRINT U$
3190 HTAB HG: PRINT L$:
3397 :
3398 REM UEBERPRUEFE OB DER BUCHSTABE IM WORT VORHANDEN IST
3399 :
3400 REM ERSETZE ALLE VORKOMMENDEN BUCHSTABEN IN C$ (HINWEISE) DES RATEVERS
      UCHS, WENN ES WELCHE GIBT
3410 REM ENTSPRECHEND DEN BEGRENZUNGEN DER TEXTFUNKTIONEN WIRD C$ BUCHSTABE
      FUER BUCHSTABE WIEDER AUFGEBAUT
3420 Z1$ = ""
3430 FOR J = 1 TO LEN(SW$)
3440 Z2$ = MID$(SW$,J,1): REM NAECHSTER UNBEKANNTER BUCHSTABE
3450 IF Z$ = Z2$ THEN Z1$ = Z1$ + Z$: REM EIN UNBEKANNTER BUCHSTABE WIRD GE
      RATEN
3460 IF Z$ < > Z2$ THEN Z1$ = Z1$ + MID$(C$,J,1): REM FALSCHER VERSUCH,
      BENUTZE INFORMATIONEN FUER HINWEISE
3470 NEXT
3480 IF Z1$ = C$ THEN 4000: REM ANHALTSPUNKTE AENDERN SICH NICHT,FALSCH GER
      ATEN
3490 C$ = Z1$: REM RICHTIG GERATEN;ERNEUERE HINWEISE UND DRUCKE AUS (EINGANG
      FUER RICHTIG GERATENES WORT)
3500 VTAB VC: HTAB HC: PRINT C$:
3510 GOTO 5000
3997 :
3998 REM ** FALSCH GERATEN **
3999 :
4000 J = NB

```

```

4010 FOR K = 1 TO 8: REM LETZTER BLOCK BLINKT
4020 J = NB
4030 GOSUB 8300
4040 YP = 2: GOSUB 11400: REM PAUSE
4050 J = NB
4060 GOSUB 8400: REM LOESCHE LETZTEN BLOCK
4070 NEXT
4100 NB = NB - 1: REM EIN BLOCK WENIGER
4110 IF NB > 0 THEN 3000: REM UEBRIGGEBLIEBENE BLOCKS
4120 GOTO 6200
4597 :
4598 REM * FEHLER *
4599 :
4600 Z$ = "DIESER BUCHSTABE WURDE SCHON VERSUCHT": REM BEREITS GERATEN
4610 GOTO 4900
4700 Z$ = "NUR A BIS Z VERWENDEN": REM KEIN ALPHABETISCHES ZEICHEN
4891 :
4892 REM * DRUCKE FEHLERMELDUNG UND PASUE *
4893 REM EINGANG: Z$ AUSZUDRUCKENDER HINWEIS
4899 :
4900 VTAB 22: HTAB 20 - LEN (Z$) / 2: REM MITTELPUNKT
4910 INVERSE : PRINT Z$;: NORMAL
4920 YP = 60: GOSUB 11400: REM PAUSE
4930 HTAB 1: PRINT LEFT$ (BL$,40): REM LOESCHE FEHLERZEILE
4940 GOTO 3000: REM NAECHSTER VERSUCH
4997 :
4998 REM ** RICHTIGER VERSUCH **
4999 :
5000 IF C$ = SW$ THEN 5100: REM ** RICHTIGER VERSUCH **
5010 REM RICHTIGER BUCHSTABE ABER WORT NOCH NICHT GERATEN
5020 W1 = 0:W2 = 1:W3 = 0:W4 = 10:W5 = - 1:W6 = 50:W7 = 2: GOSUB 13400
5030 FOR K = 1 TO 15
5040 J = INT (NB * RND (1)) + 1: REM NEHME EINEN ZUFAELLIGEN BLOCK
5050 GOSUB 8300: REM UND WECHSLE DIE FARBE
5060 NEXT
5070 GOTO 3000: REM NAECHSTER VERSUCH
5100 WD = 100:Z$ = SW$: GOSUB 13300: REM * UNBEKANNTES WORT GEFUNDEN *
5110 FOR K = 1 TO 4
5120 FOR J = 1 TO NB: GOSUB 8400: NEXT
5130 FOR J = 1 TO NB: GOSUB 8300: NEXT
5140 NEXT
5150 GOTO 6000: REM NOCH EINMAL?
5997 :
5998 REM ** NOCH EINMAL? **
5999 :
6000 POKE 34,22: HOME : REM LOESCHE DIE UNTEREN 2 ZEILEN
6010 VTAB 24: HTAB 1
6020 PRINT "NOCH EINMAL SPIELEN (J OR N)? ";
6030 YF$ = " ": GOSUB 11200: REM J/N
6040 ON Z$ + 2 GOTO 6100,6000,2000,6100: REM ESC,UNGUELTIG, J, N
6100 PRINT : PRINT
6110 PRINT "VIELEN DANK FUER'S MITSPIELEN.";
6120 END
6197 :
6198 REM * HOERE AUF *
6199 :
6200 VTAB 22: HTAB HC - 14
6210 INVERSE : PRINT "DAS WORD WAR : ";SW$: NORMAL
6220 GOTO 6000
6797 :
6798 REM * KEINE WEITEREN WORTE *

```

```

6799 :
6800 HOME
6810 PRINT
6820 PRINT "WIR HABEN ALLE UNBEKANNTEN WOERTER BENUTZT"
6830 GOTO 6100
7997 :
7998 REM  ** BLOCK PROZEDUREN **
7999 :
8000 POKE 34,20: HOME : REM  ** SETZE BLOCKS AN STELLE ** (LOESCHE TEXTZEILE
N)
8010 COLOR= 0: FOR Z = BT TO 39: HLIN 0,39 AT Z: NEXT : REM  LOESCHE BLOCK B
EREICHE
8020 FOR J = 1 TO NB
8030 GOSUB 8100: REM  LOKALISIERE UND DRUCKE NAECHSTEN BLOCK AUS
8040 NEXT
8050 RETURN
8100 H = INT ((40 - BH) * RND (1)): REM  * LOKALISIERE UND DRUCKE BLOCK UNT
ERHALB DES TITELS AUS *
8110 REM  J = BLOCK #
8120 REM  DER BLOCK DARF NICHT ZU NAHE AN DEN BLOECKEN 1,...,J-1 SEIN
8130 V = INT ((39 - BV - BT) * RND (1)) + BT
8140 IF J = 1 THEN 8210: REM  ERSTER BLOCK LOKALISIERT
8150 FOR Z = 1 TO J - 1
8160 IF ABS (BK%(Z,0) - H) > BH OR ABS (BK%(Z,1) - V) > BV THEN 8200: REM
NICHT ZU NAHE
8170 H = INT ((40 - BH) * RND (1)): REM  MACHE NEUE LOKALISIERUNG
8180 V = INT ((39 - BV - BT) * RND (1)) + BT
8190 Z = 0: REM  UND UEBERPRUEFE WIEDER OB ZU NAHE
8200 NEXT
8210 BK%(J,0) = H: REM  H-POS VON BLOCK J
8220 BK%(J,1) = V: REM  V-POS
8230 REM  DRUCKE EINEN BLOCK AUS
8300 C = INT (15 * RND (1)) + 1: REM  * DRUCKE EINEN BLOCK *
8310 REM  J= BLOCK #
8320 COLOR= C: REM  FESTER BLOCK(EINGANG FUER LOESCHEN EINES BLOCKS)
8330 H = BK%(J,0): REM  H-POS
8340 V = BK%(J,1): REM  V-POS
8350 IF SCRN( H,V) = C THEN 8300: REM  NEHME ANDERE FARBE WENN GLEICHE WIE
BEVOR
8360 FOR Z = 1 TO BH
8370 VLIN V,V + BV - 1 AT H + Z - 1
8380 NEXT
8390 RETURN
8400 C = 0: REM  * LOESCHE EINEN BLOCK *
8410 REM  J = BLOCK #
8420 GOTO 8320: REM  LESE CODE
8500 J = INT (NB * RND (1)) + 1: REM  * BEWEGE EINEN BLOCK *
8510 REM  LOESCHE EINEN ZUFALLIGEN BLOCK, UEBERSCHREIBE BLOCK MIT DEM LETZTE
N BLOCK
8520 REM  DANN LOKALIOSIERE UND DRUCKE EINEN NEUEN BLOCK AUS
8530 GOSUB 8400: REM  LOESCHE DEN GEWAELHTEN BLOCK
8540 BK%(J,0) = BK%(NB,0): REM  UEBERSCHREIBE BLOCK J MIT DEM LETZTEN BLOCK
8550 BK%(J,1) = BK%(NB,1)
8560 J = NB
8570 GOTO 8100: REM  LOKALISIERE UND DRUCKE NEUEN BLOCK
8997 :
8998 REM  ** INFORMATIONEN **
8999 :
9000 TEXT : HOME
9010 PRINT "BLOCKOUT IST EIN SPIEL ZUM RATEN          VON WOERTERN."
9020 PRINT

```

```

9030 PRINT
9040 PRINT "DER COMPUTER WAEHLT EIN UNBEKANNTES "
9050 PRINT "WORT UND DRUCKT EINEN PUNKT FUER JEDEN"
9060 PRINT "BUCHSTABEN AUS. (EIN 6-BUCHSTABEN WORT ERHAELT DESHALB 6 PUNKTE.
)"
9070 PRINT
9080 PRINT "VERSUCHE DIE BUCHSTABEN ZU ERRATEN."
9090 PRINT
9100 PRINT "JEDER RICHTIGE BUCHSTABE WIRD IN DEM"
9110 PRINT "UNBEKANNTEN WORT GEZEIGT. BEI JEDEM "
9120 PRINT "FALSCHEN VERSUCH VERSCHWINDET EINER DER BLOCKS VOM BILDSCHIRM."
9130 PRINT
9140 PRINT "WIR BEGINNEN MIT ";NG;" BLOCKS.                VERSUCHE DAS UNBEKAN
NTE WORT ZU RATEN,"
9150 PRINT "BEVOR ALLE BLOCKS VERSCHWUNDEN SIND. "
9160 PRINT
9170 PRINT "DRUECKE ";; INVERSE : PRINT "ESC";: NORMAL : PRINT " FUER SPIELENDE."
9180 PRINT
9190 PRINT "MIT RETURN GEHT ES WEITER...";
9200 GOSUB 11500: REM  WARTEN AUF TASTENDRUCK
9210 IF Z% = - 1 THEN G100: REM  ESC
9300 GR : HOME : REM  TITEL
9310 X% = "BLOCK":XC(1) = INT (15 * RND (1)) + 1:XV = 0: GOSUB 15300: REM  TITLE
9320 X% = "OUT":XV = XV + XB + 1: GOSUB 15300
9330 BT = XV + XB + 1: REM  OBERSTER BLOCK BEREICH
9340 RETURN
9497 :
9498 REM  ** SETZE BLOCKS **
9499 :
9500 NB = NG: REM  ANZAHL DER BLOECKE
9510 GOSUB 8000: REM  SET UP BLOCKS
9520 VTAB VG: HTAB HG - 7: PRINT "RATE :";
9530 VTAB VC: HTAB HC: PRINT C%
9540 PRINT
9550 PRINT "BENUTZT: "U$
9560 PRINT "REST:  "L$;
9570 RETURN
50997 :
50998 REM  ** WOERTER, WERDEN MIT "END" BEENDET
50999 :
51000 DATA  "STRATEGIE","QUADER" ,"AERGERN  ","ZITTERPAPPEL"
51010 DATA  "TABLETTE","FLECHTE","ZAEHNE","ERSTAUNEN"
51020 DATA  "RATESPIEL","EINLAGE","VERAENDERN","SCHMEICHLER"
51030 DATA  "UNENDLICH","PIANISSIMO","FUSSBALLTOR","GRAZIL"
51040 DATA  "SAUERSTOFF","VERWELKEN","RINSENWEISHEIT","GEHIRN"
51050 DATA  "TAPFERKEIT","BOHRER","PFERD"
51090 DATA  "END"
60000 :
60010 REM  * COPYRIGHT 1981 BY HOWARD FRANKLIN, PALO ALTO, CA *
60020 :

```

Verbinden Sie dieses Programm mit EINGABE EINHEIT, KLANG EINHEIT (löschen Sie die Zeilen 18000 bis 19999) und BILD EINHEIT. Speichern Sie es als BLOCKOUT und tippen Sie RUN.

BLOCKOUT nutzt die Möglichkeiten der LO-RES-Grafik. Die Wörter werden aus einer Auflistung entnommen, die in den DATA-Anweisungen von Zeile 51000 bis 51090 niedergeschrieben ist. Es können Wörter geändert und neue Ausdrücke hinzugefügt werden. Die Anzahl der zur Auswahl stehen-

den Wörter beträgt maximal 50 (siehe Zeile 1000). In Zeile 51090 müssen wir ein Zeichen (flag) setzen. Dieses Zeichen teilt dem Rechner mit, daß keine weiteren Wörter folgen. In der hier vorgestellten Form sind 10 Fehlversuche erlaubt.

Frage 5: Wie ist das Programm zu verändern, damit maximal sechs Fehlversuche erlaubt sind?

Beachten Sie, daß BLOCKOUT die ESC-Funktion in der gleichen Art verwendet, wie dies im letzten Kapitel vereinbart wurde. Mit ESC und anschließend RETURN kann der Benutzer das Spiel beenden und das unbekannte Wort sehen. Die farbigen Quadrate werden zufällig auf dem Bildschirm verteilt. Auch die Farben werden per Zufallsfunktion gewählt.

Frage 6: Wie müssen Sie das Programm ändern, damit alle Quadrate die Farbe Orange erhalten.

Schauen wir uns den Programmteil, beginnend mit Zeile 8100 an. Diese Prozedur überprüft sorgfältig den Abstand zwischen den einzelnen Farbquadraten, die nicht zu nahe beisammen platziert werden sollten, damit eine ansprechende grafische Bildgestaltung erreicht wird.

Eine falsche Antwort bewirkt, daß eines der Farbquadrate einige Male blinkt und anschließend verschwindet.

Frage 7: Welcher Programmteil ruft das Verschwinden der Farbquadrate hervor?

In vielen grafischen Spielen wird das Vier-Zeilen-Textfenster am unteren Bildschirmrand nicht optimal ausgenutzt. Oft fallen wichtige Fragen aus dem sichtbaren Bereich heraus. Wichtige Informationen für den Spieler, der vielleicht vergessen hat, was zu tun ist, gehen verloren.

Bei BLOCKOUT bleiben alle Anhaltspunkte, wie bereits benutzte Buchstaben und noch verbleibende Buchstaben, sichtbar. Nur die Frage wird erneuert. Dies ist eine Möglichkeit, das Textfenster effektiv zu nutzen. Allgemein sollte das grafische Bild so gestaltet werden, daß alle wichtigen Bestandteile während des ganzen Spiels auf dem Bildschirm sichtbar bleiben.

Haben Sie bemerkt, daß beim Spielen von BLOCKOUT jede richtige Antwort ein akustisches Signal hervorruft? Bei falschen Antworten ändert sich zwar der Bildschirm, aber es ertönt kein akustisches Signal. Das Verstärken richtiger Antworten und, wenn möglich, das Ignorieren falscher Antworten ist eine empfehlenswerte Technik für das Schreiben von Lehr- und Lernprogrammen.

Schauen Sie sich sorgfältig den Programmteil TITELBILD ab Zeile 1400 an. Man könnte meinen, die grafischen Aktivitäten auf dem Bildschirm erforderten viele Programmbefehle. Da unser Programm BILD EINHEIT jedoch so geschickt konstruiert ist, benötigen wir nur wenig Programmieraufwand zur Herstellung eines attraktiven Bildes. Schauen Sie sich auch die anderen Spielprogramme dieses Buches an. Beachten Sie, wie wenig Programmierzeilen für die Herstellung eines schönen Titelbildes erforderlich sind!

Match

MATCH ist ein Ein-Personen-Spiel, das auch mehrere Personen abwechselnd spielen können. Das Ziel des Spiels ist es, zusammengehörige Wortpaare zu bilden. Der Spieler kann jederzeit das Spiel durch Drücken von ESC RETURN beenden. Als wir MATCH entworfen haben, entschieden wir uns dafür, nach Beendigung des Spiels die richtigen Antworten nicht anzugeben. Dies ist auch nicht notwendig, denn durch Ausscheiden der falschen Wörter kann der Spieler immer die richtige Lösung finden. Eine der positiven Folgen der ESC-Vereinbarung in EINGABE EINHEIT ist die ständige Kontrolle über die Funktion der ESC-Taste.

Hier haben wir die Funktion so festgelegt, daß keine richtigen Ergebnisse gezeigt werden:

```
10 REM    ...MATCH-EINGABE KLANG BILD
11 :
997 :
998 REM  ** INITIALISIERUNG **
999 :
1000 DIM L$(50),R$(50),L%(50),R%(30)
1010 BL$ = ""
1020 FOR J = 1 TO 40
1030 BL$ = BL$ + " "
1040 NEXT
1100 Z = 51000: GOSUB 19000: REM  SETZE READ DATA ZEIGER
1110 READ HL,HR,C$
1120 Z = HR - HL - LEN (C$): IF Z < 2 THEN HOME : PRINT "DAS VERBINDUNGSWORT
    IST ZU LANG": END
1130 IF Z / 2 < > INT (Z / 2) THEN HR = HR - 1: REM  GLEICHE ZAHL VON LEER
    STELLEN AUF BEIDEN SEITEN
1140 HC = INT ((HL + HR) / 2)
1150 VO = 2
1160 VS = 2
1200 NP = 0: REM  INITIALISIERE WORTPAARE
1210 READ Z$,Z1$
1220 IF Z$ = "END" OR Z1$ = "END" THEN 1300: REM  KEINE WEITEREN WORTPAARE
1230 NP = NP + 1: REM  WEITERE WORTPAARE
1240 L$(NP) = Z$
1250 R$(NP) = Z1$
1260 GOTO 1210
1300 NR = 8: REM  8 REIHEN
1397 :
1398 REM  * TITELBILD *
1399 :
1400 BR : HOME : COLOR= 13: GOSUB 15500: REM  GELBER HINTERGRUND
1410 X$ = "MATCH":XV = 10:XC(1) = 2: GOSUB 15300
1420 X$ = "W" + CHR$ (1) + CHR$ (2) + "CH":XV = 40 - XV - XB:XC(1) = 7: GOSUB
    15300: REM  SPIEGELBILD
1430 VTAB 23: HTAB 8: PRINT "MIT RETURN GEHT ES WEITER..";
1440 GOSUB 11500: REM  WARTEN AUF TASTENDRUCK
1450 IF Z% = - 1 THEN END : REM  ESC
1900 GOSUB 9000: REM  INFORMATIONEN
1997 :
1998 REM  ** INITIALISIERUNG FÜR NÄCHSTES SPIEL **
1999 :
2000 FOR J = 1 TO NP
```



```

2010 L%(J) = - J: REM   FLAG FUER ANFANGSAUSDRUCK
2020 NEXT
2030 N = NP: REM   VERTAUSCHE ALLE PAARE
2040 GOSUB 2900
2050 N = NR: REM   BENUTZE ERSTE NR
2060 FOR J = 1 TO N
2070 R%(J) = L%(J): REM   COPY TO R%(1,...,NR)
2080 NEXT
2090 GOSUB 2900: REM   VERTAUSCHE ALLE PAARE DER LINKEN SEITE
2200 GOSUB 9500: REM   INITIALISIERE BILDSCHIRMAUSDRUCK
2210 GOTO 3000
2900 FOR Z = N TO 2 STEP - 1: REM   VERTAUSCHE L%(1,...,N)
2910 Z% = Z * RND (1) + 1
2920 Z1 = L%(Z)
2930 L%(Z) = L%(Z%)
2940 L%(Z%) = Z1
2950 NEXT
2960 RETURN
2997 :
2998 REM   ** WAEHLE NAECHSTES PAAR **
2999 :
3000 FOR Z = 1 TO N
3010 VTAB V0 + VS * (NR - N + Z): REM   MACHE MENUE
3020 HTAB HC: PRINT Z;
3030 NEXT
3100 QP$ = "WAEHLE EIN WORT DER LINKEN SEITE:  "
3110 GOSUB 3500
3120 P1 = J
3130 INVERSE : REM   DRUCKE LINKEN TEXT INVERS AUS
3140 GOSUB 8600
3150 NORMAL
3200 QP$ = "NUN WAEHLE EIN WORT DER RECHTEN SEITE:"
3210 GOSUB 3500
3220 P2 = J
3230 INVERSE : REM   DRUCKE RECHTEN TEXT INVERS AUS
3240 GOSUB 8800
3250 NORMAL
3300 FOR Z = 1 TO N: REM   LOESCHE MENUE
3310 VTAB V0 + VS * (NR - N + Z): REM   V-POS
3320 HTAB HC: REM   H-POS
3330 PRINT " ";
3340 NEXT
3400 IF L%(P1) < > R%(P2) THEN 4000: REM   FALSCH KOMBINATION
3410 GOTO 5000: REM   RICHTIG
3491 :
3492 REM   * ALLGEMEINER CODE FUER WAHL EINES TEXTES *
3493 REM   EINGANG: QP$ SOFORT
3494 REM   AUSGANG: J   WAHL
3499 :
3500 VTAB 22: HTAB 1: PRINT QP$;
3510 YL = 1: YH = N: YW = 1: GOSUB 11000
3520 IF Z% = - 1 THEN 6100
3530 IF Z% = 0 THEN 3800: REM   UNGUELTIG
3540 J = Z
3550 HTAB 1: PRINT LEFT$ (BL$,39);
3560 RETURN
3800 Q$ = "WAEHLE EINE ZAHL ZWISCHEN 1 UND " + STR$ (N)
3810 P = 80
3820 GOSUB 3900
3830 GOTO 3500: REM   VERSUCHE NOCH EINMAL
3891 :

```

```

3892 REM * DRUCKE HINWEIS ZEILE *
3893 REM EINGANG: Q$ AUSZUDRUCKENDER TEXT
3894 REM P PAUSE
3899 :
3900 VTAB 24: HTAB 20 - INT ( LEN (Q$) / 2)
3910 INVERSE : PRINT Q$;: NORMAL
3920 YP = P: GOSUB 11400: REM PAUSE
3930 HTAB 1: PRINT LEFT$ (BL$,39);
3940 RETURN
3997 :
3998 REM ** FALSCH KOMBINATION **
3999 :
4000 YP = 10
4010 IF P1 = P2 THEN 4400: REM LINKS,RECHTS VERSCHIEBEN
4020 IF P2 < P1 THEN 4200: REM BEWEGE ERST RECHTE SEITE NACH UNTEN
4030 UD = 1: REM BEWEGE LINKE SEITE EINS NACH UNTEN
4040 GOSUB 8000
4050 GOTO 4010
4200 UD = 1: REM BEWEGE RECHTE SEITE EINS NACH UNTEN
4210 GOSUB 8200
4220 GOTO 4010
4400 IF P1 = N THEN 4600: REM LINKE,RECHTE SEITE AM BODEN
4410 UD = 1: REM BEWEGE BEIDE SEITEN
4420 GOSUB 8400
4430 GOTO 4400
4600 W1 = 0:W2 = 5:W3 = 0:W4 = 10:W5 = 1:W7 = 1: GOSUB 13400
4610 Q$ = "** FALSCH KOMBINATION **"
4620 P = 60
4630 GOSUB 3900
4640 J = P1
4650 GOSUB 8600: REM DRUCKE NORMAL
4660 GOSUB 8800
4670 GOTO 3000
4997 :
4998 REM ** RICHTIGE KOMBINATION **
4999 :
5000 YP = 10
5010 IF P1 = P2 THEN 5400: REM LINKE,RECHTE SEITE VERSCHIEBEN
5020 IF P2 > P1 THEN 5200: REM BEWEGE RECHTE SEITE ZUERST
5030 UD = - 1: REM BEWEGE LINKE SEITE
5040 GOSUB 8000
5050 GOTO 5010
5200 UD = - 1: REM BEWEGE RECHTE SEITE
5210 GOSUB 8200
5220 GOTO 5010
5400 IF P1 = 1 THEN 5600:LINKE,RECHTESEITEOBEN
5410 UD = - 1: REM MOVE BOTH SIDES
5420 GOSUB 8400
5430 GOTO 5400
5600 W1 = 0:W2 = 5:W3 = 0:W4 = 10:W5 = - 1:W7 = 1: GOSUB 13400
5610 VTAB V0 + VS * (NR - N + 1): REM LEFT,RIGHT ON TOP LINE
5620 HTAB HC - INT ( LEN (C$) / 2)
5630 INVERSE : PRINT C$;: NORMAL
5640 Z$ = L$(L$(1))
5650 Z1$ = R$(R$(1))
5700 FOR J = HL + 1 TO HC - INT ( LEN (C$) / 2) - 1: REM BEWEGE WOERTER ZU
SAMMEN
5710 HTAB J - LEN (Z$) - 1: PRINT " ";: INVERSE : PRINT Z$;: NORMAL
5720 HTAB HR + HL - J: INVERSE : PRINT Z1$;: NORMAL : PRINT " ";
5730 NEXT

```

```

5800 Q$ = "DAS IST RICHTIG!!!"
5810 P = 60
5820 GOSUB 3900
5830 N = N - 1: REM   EIN PAAR WENIGER
5840 IF N = 0 THEN 5900: REM   ALLE GEFUNDEN
5850 FOR J = 1 TO N: REM   BEWEGE LISTE ABWAERTS
5860 L%(J) = L%(J + 1)
5870 R%(J) = R%(J + 1)
5880 NEXT
5890 GOTO 3000
5900 VTAB 21: HTAB 7: PRINT "DU HAST ALLE WORTPAARE GEFUNDEN!!"
5997 :
5998 REM   ** NOCHMAL? **
5999 :
6000 HTAB 1: VTAB 24
6010 PRINT "NOCH EINMAL (J OR N)? ";
6020 GOSUB 11200: REM   J/N
6030 ON Z% + 2 GOTO 6100,6000,2000,6100: REM   ESC,UNGUELTIG,J,N
6100 PRINT : PRINT
6110 PRINT "VIELEN DANK FUER'S MITSPIELEN.";
6120 END
7997 :
7998 REM   ** BEWEGUNGS PROZEDUREN **
7999 :
8000 Z = L%(P1): REM   BEWEGE LINKE SEITE AUF/AB
8010 REM   UD = -1 AUF, +1 AB
8020 L%(P1) = L%(P1 + UD)
8030 L%(P1 + UD) = Z
8100 J = P1: REM   DRUCKE NEUES LINKES J
8110 GOSUB 8700: REM   LEERZEICHEN
8120 GOSUB 8600: REM   DANN DRUCKE
8130 J = P1 + UD:P1 = J: REM   BEWEGE WAHL
8140 GOSUB 8700: REM   LEERZEICHEN
8150 INVERSE
8160 GOSUB 8600: REM   DRUCKE INVERS
8170 NORMAL
8180 GOSUB 11400: REM   PAUSE
8190 RETURN
8200 Z = R%(P2): REM   BEWEGE RECHTE SEITE AUF/AB
8210 REM   UD = -1 AUF, +1 AB
8220 R%(P2) = R%(P2 + UD)
8230 R%(P2 + UD) = Z
8300 J = P2: REM   DRUCKE NEUES RECHTES J
8310 GOSUB 8900: REM   LEERZEICHEN
8320 GOSUB 8800: REM   DANN DRUCKE
8330 J = P2 + UD:P2 = J: REM   BEWEGE WAHL
8340 GOSUB 8900: REM   LEERZEICHEN
8350 INVERSE
8360 GOSUB 8800: REM   DANN DRUCKE INVERS
8370 NORMAL
8380 GOSUB 11400: REM   PAUSE
8390 RETURN
8400 Z = L%(P1): REM   BEWEGE BEIDE SEITEN
8410 REM   UD = -1 AUF, +1 AB
8420 L%(P1) = L%(P1 + UD)
8430 L%(P1 + UD) = Z
8440 Z = R%(P1)
8450 R%(P1) = R%(P1 + UD)
8460 R%(P1 + UD) = Z
8470 J = P1

```

```

8480 GOSUB 8700
8490 GOSUB 8900
8500 GOSUB 8600
8510 GOSUB 8800
8520 J = P1 + UD:P1 = J: REM BEWEGE REIHE
8530 GOSUB 8700
8540 GOSUB 8900
8550 INVERSE
8560 GOSUB 8600
8570 GOSUB 8800
8580 NORMAL
8590 GOTO 11400: REM PAUSE
8597 :
8598 REM * DRUCKE PROZEDUREN *
8599 :
8600 Z$ = L$(L%(J)): REM DRUCKE LINKE SEITE IN REIHE J
8610 HTAB HL - LEN (Z$): REM H-POS
8620 GOTO 8820
8700 Z$ = LEFT$ (BL$,HL - 1): REM LEERZEICHEN LINKER TEXT IN REIHE J
8710 GOTO 8610
8800 Z$ = R$(R%(J)): REM DRUCKE RECHTE SEITE IN REIHE J
8810 HTAB HR: REM H-POS
8820 VTAB VO + VS * (NR - N + J): REM EINGANG FUER LINKEN TEXT
8830 PRINT Z$;
8840 RETURN
8900 Z$ = LEFT$ (BL$,40 - HR): REM LEERZEICHEN RECHTER TEXT IN REIHE J
8910 GOTO 8810
8997 :
8998 REM ** INFORMATIONEN **
8999 :
9000 TEXT : HOME
9010 VTAB 4
9020 HTAB 14: PRINT "*** MATCH ***"
9030 PRINT
9040 PRINT
9050 PRINT "IN DIESEM SPIEL WOLLEN WIR WORTPAARE BILDEN."
9060 PRINT
9070 PRINT "WELCHES WORT AUF DER LINKEN SEITE"
9080 PRINT "PASST ZU EINEM WORT AUF DER RECHTEN SEITE?"
9090 PRINT "VERBINDE DIE WOERTER DURCH ANGABE IHRER NUMMERN."
9100 PRINT
9110 PRINT "WENN DU ALLE WORTPAARE GEFUNDEN HAST, IST DAS SPIEL GEWONNEN."
9120 VTAB 20
9130 PRINT "DRUECKE "; INVERSE : PRINT "ESC"; NORMAL : PRINT " FUER SPIELEN
DE."
9140 PRINT
9150 PRINT "MIT RETURN GEHT ES WEITER..";
9160 GOSUB 11500: REM WARTE AUF TASTENDRUCK
9170 IF Z% = - 1 THEN 6100: REM ESC
9180 RETURN
9497 :
9498 REM * INITIALISIERE AUSDRUCK *
9499 :
9500 HOME
9510 HTAB HC - 2: INVERSE : PRINT "VERBINDE": NORMAL
9520 FOR K = 1 TO NR: REM DRUCKE PAARE ZEFAELLIG AUS
9530 J = INT (NR * RND (1)) + 1: REM DRUCKE NEUES LINKES PAAR
9540 IF L%(J) > 0 THEN 9530: REM SCHON AUSGEDRUCKT,VERSUCHE NOCHMAL
9550 L%(J) = - L%(J): REM FLAG WENN GEDRUCKT
9560 GOSUB 8600: REM DRUCKE LINKES PAAR

```

```

9570 J = INT (NR * RND (1)) + 1: REM   DRUCKE NEUES RECHTES PAAR
9580 IF R%(J) > 0 THEN 9570: REM   SCHON GEDRUCKT, VERSUCHE NOCHMAL
9590 R%(J) = - R%(J): REM   FLAG WENN GEDRUCKT
9600 GOSUB 8800: REM   DDRUCKE RECHTES PAAR
9610 NEXT
9620 RETURN
20100 DATA 5,7: REM   UMGEDREHTES A
20110 DATA "1 1"
20120 DATA "1 1"
20130 DATA "11111"
20140 DATA "1 1"
20150 DATA "1 1"
20160 DATA " 1 1"
20170 DATA " 1"
20180 DATA "-1"
20200 DATA 5,7: REM   UMGEDREHTES T
20210 DATA " 1"
20220 DATA " 1"
20230 DATA " 1"
20240 DATA " 1"
20250 DATA " 1"
20260 DATA " 1"
20270 DATA "11111"
20280 DATA "-1"
49997 :
49998 REM   ** VARIABLEN VEREINBARUNGEN **
49999 :
50000 REM   L$(J) LINKER TEXT DES PAARES (RECHTS ANGEPASST)
50010 REM   R$(J) RECHTER TEXT DES PAARES (LINKS ANGEPASST)
50020 REM   LZ(J) LINKER TEXT IN J-TER REIHE
50030 REM   RZ(J) RECHTER TEXT IN J-TER REIHE
50040 REM   NP ZAHL DER PAARE
50050 REM   NR ZAHL DER REIHEN
50060 REM   N   ZEHL DER UEBRIGEN REIHEN
50070 REM   HL   H-POS + 1 DES RECHTEN RANDES DER LINKEN SPALTE
50080 REM   HR   H-POS DES LINKEN RANDES DER RECHTEN SPALTE
50090 REM   HC   H-POS DER MITTE
50100 REM   VO   V-POS VON "REIHE 0"
50110 REM   VS   ZAHL DER VERTIKELN LEERSTELLEN ZWISCHEN DEN REIHEN
50120 REM   BL$  LEERZEICHEN
50130 REM   P1   ERSTE WAHL
50140 REM   P2   ZWEITE WAHL
50990 :
50991 :
50992 REM   ** WORTPAARE DATEN BEGINNEN IN ZEILE 51000 **
50993 :
50994 REM   H-POS + 1 DES RECHTEN RANDES DER LINKEN SPALTE
50995 REM   H-POS DES LINKEN RANDES DER RECHTEN SPALTE
50996 REM   VERBUNDUNGSWORT (MUSS ZWISCHEN DIE SPALTEN PASSEN)
50997 REM   PAARE 1,...,NP
50998 REM   END,END
50999 :
51000 DATA 14,27,"BEDEUTET"
51010 DATA "RESIGNATION","AUFGABE" ,"GRAVITATION","SCHWERKRAFT","STILL" ,"
RUHIG"
51020 DATA "HEKTISCH","TURBULENT","HEISSPORN","HITZKOPF","PERFEKT" ,"FEH
LERFREI"
51030 DATA "TAKTLOS" ,"UNHOFELICH","BRANCHE" ,"BERUFSZWEIG","MONARCH","
HERRSCHER"
51040 DATA "TEENAGER","JUGENDLICHER","DISTANZ","ABSTAND","ABWEHREN" ,"VER
TEIDIGEN"

```

```

51050 DATA "ANGREIFEN", "STUERMEN", "SCHWINGEN" , "VIBRIEREN", "VERTIKAL", "SEN
KRECHT"
51060 DATA "HORIZONTAL", "WAAGERECHT", "BELEGEN", "BEWEISEN", "KALKULIEREN", "RE
CHNEN"
51070 DATA "WAGNIS", "RISIKO", "VEGETARISCH", "PFLANZLICH" , "DREHUNG" , "ROT
ATION"
51080 DATA "VERDAMPFEN", "VERDUNSTEN", "KOSTBAR", "WERTVOLL", "NUETZLICH", "BRAU
CHBAR"
51090 DATA "UNSINN", "NONSENS", "REGEL" , "NORM" , "AUFSTEHEN", "ERHEBEN
"
51100 DATA "AUFSTAND" , "REVOLUTION", "OZEAN" , "MEER" , "INSEL" , "EILAND"
51110 DATA "VERWUNDET", "VERLETZT" , "PRUEFEN" , "ERPROBEN", "SPION" , "
AGENT"
51120 DATA "HUENE" , "RIESE" , "GESETZLICH", "LEGAL" , "THESE" , "LEITSATZ"
51130 DATA "MOTIV" , "ABSICHT" , "COMPUTER", "RECHNER", "POET" , "DICHT
E"
51999 DATA "END", "END"
60000 :
60010 REM * COPYRIGHT 1981 BY HOWARD FRANKLIN, PALO ALTO, CA *
60020 :

```

Verbinden Sie das Programm mit EINGABE EINHEIT, KLANG EINHEIT (löschen Sie die Zeilen 18000 bis 18999) und BILD EINHEIT. Speichern Sie es als MATCH und starten Sie es.

MATCH benutzt das ganze Bildschirmformat und die inverse Darstellung. Obwohl es auf der Darstellung von Texten basiert, ist der Bildschirmausdruck sehr schön anzuschauen. MATCH wurde so entworfen, daß es dem Spieler möglichst alle nötigen Hilfen gibt. Die wichtigen Informationen bleiben jederzeit sichtbar. Die Schrift bleibt immer innerhalb des Textfensters. Schließlich benutzt das Programm sehr raffinierte Techniken, um Texte zu verschieben. Dadurch ergibt sich eine lebendige und abwechslungsreiche Textgestaltung.

MATCH ist ein nettes Spiel und ein gutes Beispiel für Lernprogramme, die auch Töne und Klänge zur Unterstützung verwenden. Es werden hilfreiche Fehlermeldungen gegeben. Die Reaktion auf die richtigen Antworten ist viel stärker und deutlicher als auf falsche. Die gefundenen Wortpaare bleiben weiter sichtbar und heben so die richtigen Antworten hervor. (Falsche Wortpaare rutschen zum unteren Bildschirmrand.)

MATCH nutzt den Bildschirm in sehr effizienter Weise. Das zentrale Bild ist sowohl attraktiv anzusehen als auch platzsparend. Die Benutzung der inversen Textdarstellung hebt richtige Antworten deutlicher hervor. Das Bewegen der Textblöcke richtet die Aufmerksamkeit des Spielers auf den Monitor. Das Bewegen der zu verbindenden Wörter und das anschließende Zusammenfassen an einer Stelle besitzt einen größeren pädagogischen Wert als das Verbinden der richtigen Wortpaare durch Linien (z. B. in Lehrbüchern).

Ein weiterer Vorzug von MATCH ist folgender: Die Liste der zur Verfügung stehenden Wörter wird kontinuierlich durchnummeriert. Dadurch kann die Zahl der noch verbleibenden Wörter wiedergegeben werden. Nach jedem neuen Start des Programms werden die Wortpaare zufällig aus den DATA-Anweisungen entnommen und in einer veränderten Reihenfolge präsentiert.

Durch Änderung des Inhalts der DATA-Anweisungen kann MATCH leicht erweitert werden. Der größte Vorteil dieses Programms liegt jedoch in den vielfältigen Möglichkeiten dieses Spiels. Es können nicht nur sinnverwandte Wörter benutzt werden, sondern beliebige Texte oder auch Zahlen. Beachten Sie, daß der verbindende Text (in diesem Fall "BEDEUTET") in der DATA-Anweisung in Zeile 51000 steht. Wir können beliebige Kombinationen einsetzen: Länder und ihre Hauptstädte, Gleichungen und ihre Lösungen, Vokabeln in Englisch und Deutsch usw. In jedem Fall muß ein entsprechender Verbindungstext eingesetzt werden.

Wir zeigen zwei Beispiele, wie der Bildschirm nach Spielende aussehen kann:

ITALIEN	HAUPTSTADT	ROM
FRANKREICH	HAUPTSTADT	PARIS
ENGLAND	HAUPTSTADT	LONDON
SPANIEN	HAUPTSTADT	MADRID
SCHWEIZ	HAUPTSTADT	BERN
BELGIEN	HAUPTSTADT	BRUESSEL
SCHWEDEN	HAUPTSTADT	STOCKHOLM
FOHLEN	IST EIN JUNGES	PFERD
LAMM	IST EIN JUNGES	SCHAF
KITZ	IST EIN JUNGES	REH
KUEKEN	IST EIN JUNGES	HUHN
KALB	IST EIN JUNGES	RIND
FERKEL	IST EIN JUNGES	SCHWEIN

Um diese Wortpaare verwenden zu können, werden die DATA-Anweisungen in den Zeilen 51010 bis 51999 durch neue Anweisungen ersetzt, die die Wortpaare enthalten.

Beispiel: 51010 DATA "ITALIEN", "ROM", "FRANKREICH", "PARIS"

Es können maximal 50 Wortpaare aufgenommen werden. Von diesen wählt das Spiel acht Zufallspaare aus. Natürlich muß auch der zentrale Text in Zeile 51000 geändert werden, damit er die neuen Wörter in der aktuellen Liste sinnvoll verbindet.

Zusammenfassung von Kapitel 6

In diesem Kapitel stellten wir drei Textspiele vor. Wir erläuterten, wie einfache Veränderungen vorgenommen werden, um die Spiele dem speziellen Teilnehmerkreis anzupassen. Die Spiele benutzen viele der in diesem Buch bereits vorgestellten Unterprogramme. Aufbau und Benutzerfreundlichkeit wurden anhand dieser Programme demonstriert. Im nächsten Kapitel zeigen wir einige interessante Spiele, die die grafischen Möglichkeiten des APPLE ausnutzen.

Lösungen zu Kapitel 6

- Frage 1: Die Länge der Antwort wird auf 3 Buchstaben begrenzt.
- Frage 2: Ändern der DATA-Anweisungen in den Zeilen 52000 bis 52050.
- Frage 3: Die Zeilen 3310 bis 3430.
- Frage 4: 5 Auswahlmöglichkeiten in Zeile 51020.
- Frage 5: 1200 NG = 6
- Frage 6: 8300 COLOR = 9
8350 :
- Frage 7: Die Zeilen 8400 bis 8420. Anschließend wird in den Zeilen 8300 bis 8390 das Farbquadrat mit schwarzer Farbe gezeichnet.

Kapitel 7

Weitere Spiele

In diesem Kapitel stellen wir drei weitere Computerspiele vor, die einige besondere Merkmale in sich vereinigen. KONZENTRATION ist ein auf Bildern basierendes Kartenspiel, das sich gut für die LO-RES-Grafik des APPLE eignet und das verwandt ist mit dem bekannten Familienspiel "Memory". Mit KONZENTRATION können wichtige Programmveränderungen sehr einfach durchgeführt werden, z.B. das Programmieren verschiedener Schwierigkeitsgrade.

STERNE ist ein Spiel zum Zahlenraten. Das Programm nutzt die hohe Rechengeschwindigkeit des Computers aus. Wir haben das Spiel in einer speziellen Version entwickelt, um die Grafikfähigkeiten des APPLE besonders hervorzuheben.

Unsere Version des SIMON-Spiels benutzt Musik, LO-RES-Grafik und fortlaufende Textzeilen.

Konzentration

KONZENTRATION ist ein Ein-Personen-Spiel, das aber auch von mehreren Spielern abwechselnd gespielt werden kann. Das Spiel dauert so lange, bis alle Kartenpaare gefunden sind. Am Ende des Spiels werden alle Karten offen (mit dem Bild nach oben) auf dem Bildschirm dargestellt.

```
10 REM   ...KONZENTRATION-EINGABE BILDER...
11 :
997 :
998 REM   ** INITIALISIERUNG **
999 :
1000 DIM PR%(12,1),CD%(24),H%(24),V%(24),CP%(6,1)
1010 MP = 12: REM   MAXIMALE ZAHL VON KARTENPAAREN
1020 MC = 6: REM   FARBPAARE
1030 REM   VERAENDERE DIE KARTENAUSDRUCK PROZEDUR IN 8300,... WENN CH ODER C
      V VERAENDERT!!
1040 CH = 4: REM   HORIZONTALE LAENGE - 1 EINER KARTE
1050 CV = 6: REM   VERTIKALE HOEHE - 1
1100 FOR J = 1 TO 24: REM   INITIALISIERE KARTENPOSITION
1110 Z% = (J - 1) / 6
```

```

1120 H%(J) = (J - 1 - 6 * Z%) * 6 + 2
1130 V%(J) = Z% * 10
1140 NEXT
1200 DIM BL$(39)
1210 BL$ = ""
1220 FOR J = 1 TO 39
1230 BL$ = BL$ + " "
1240 NEXT
1297 :
1298 REM * TITELBILD *
1299 :
1300 GR : HOME : FOR J = 1 TO 24: GOSUB 8000: NEXT : REM DRUCKE KARTEN MIT
      BILD NACH UNTEN
1310 XS = 1: XH = H%(1): XV = V%(1): X$ = "KON": XC(1) = 14: GOSUB 15400
1320 XH = H%(8): XV = V%(8): X$ = "ZEN": GOSUB 15400
1330 XH = H%(16): XV = V%(16): X$ = "TRA": GOSUB 15400
1340 XS = 2: XH = H%(20): XV = V%(20): X$ = "TID": GOSUB 15400
1350 XH = H%(23): X$ = "N": GOSUB 15400
1360 VTAB 23: HTAB 7: PRINT "MIT RETURN GEHT ES WEITER..";
1370 GOSUB 11500: REM WARTEN AUF TASTENDRUCK
1380 IF Z% = - 1 THEN END : REM ESC
1497 :
1498 REM * PARAMETER FÜR DIESES SPIEL *
1499 :
1500 NT = 4: REM ZAHL DER KARTENMOTIVE IN DIESEM SPIEL
1510 NC = 4: REM ZAHL DER FARBPAARE
1520 NP = 6: REM ZAHL DER KARTENPAARE
1530 LC$ = CHR$(ASC("A") - 1 + 2 * NP)
1600 GOSUB 9000: REM INFORMATIONEN
1997 :
1998 REM ** INITIALIZATION FOR NEXT GAME **
1999 :
2000 IF NT * NC < NP THEN TEXT : HOME : VTAB 11: HTAB 2: PRINT "NICHT GENUG
      KARTEN. ÄNDERE 1500-1520.": END
2010 N = 15
2020 FOR J = 1 TO N: REM INITIALISIERE FARBEN
2030 CD%(J) = J
2040 NEXT
2050 GOSUB 2900: REM WECHSELN DIE FARBEN
2060 FOR J = 1 TO NC: REM WÄHLE DIE ERSTEN 2*NC FARBEN
2070 CP%(J,0) = CD%(2 * J - 1)
2080 CP%(J,1) = CD%(2 * J)
2090 NEXT
2100 N = NP: REM ZAHL DER KARTENPAARE
2110 FOR J = 1 TO N: REM INITIALISIERE KARTENMOTIVE UND ANORDNUNG
2120 PR%(J,0) = NT * RND(1) + 1: REM WÄHLE MOTIV FÜR PAAR J
2130 PR%(J,1) = NC * RND(1) + 1: REM WÄHLE FARBE FÜR PAAR J
2140 IF J = 1 THEN 2200
2150 Z = 0
2160 FOR K = 1 TO J - 1: REM DAS PAAR SOLL VERSCHIEDEN VON DEM VORHERGEHEND
      EN PAAR SEIN
2170 IF PR%(J,0) = PR%(K,0) AND PR%(J,1) = PR%(K,1) THEN Z = 1: REM SETZE Z
      EICHEN FÜR GLEICHES PAAR
2180 NEXT
2190 IF Z > 0 THEN 2120: REM WÄHLE DAS PAAR NOCHMAL
2200 CD%(2 * J - 1) = J: REM ZWEI KARTEN FÜR PAAR J
2210 CD%(2 * J) = J
2220 NEXT
2230 GOSUB 9500: REM INITIALISIERE DEN AUSDRUCK VOR DEM MISCHEN DER KARTEN
2300 N = 2 * NP: REM ZAHL DER BENUTZTEN KARTEN

```

```

2310 GOSUB 2900: REM  MISCHEN DIE KARTEN
2320 TR = 0: REM  ZAHL DER VERTAUSCHUNGEN
2400 GOTO 3000
2900 FOR Z = N TO 2 STEP - 1: REM  MISCHEN CD%(1,...,N)
2910 Z% = Z * RND (1) + 1
2920 Z1 = CD%(Z): REM  VERTAUSCHEN ZWEI ELEMENTE
2930 CD%(Z) = CD%(Z%)
2940 CD%(Z%) = Z1
2950 NEXT
2960 RETURN
2997 :
2998 REM  ** WAEHLE NEUES PAAR **
2999 :
3000 QP$ = "ERSTE KARTEN: "
3010 QH = 10
3020 P1 = 0: REM  ERLAUBE JEDE GUELTIGE WAHL
3030 GOSUB 3500: REM  NIMM EINE GUELTIGE KARTEN
3040 IF J = 0 THEN 5200
3050 P1 = J: REM  ERSTE WAHL
3100 QP$ = "ZWEITE:"
3110 QH = 20
3120 GOSUB 3500: REM  WAEHLE EINE ANDERE GUELTIGE KARTEN
3130 IF J = 0 THEN 5200
3140 P2 = J: REM  ZWEITE WAHL
3150 TR = TR + 1: REM  EINE WEITERE VERTAUSCHUNG
3200 VTABLE 23: HTABLE 10: PRINT LEFT$ (BL$,30): REM  LEERE ZEILE
3210 IF CD%(P1) < > CD%(P2) THEN 4000: REM  KARTEN PASSEN NICHT
3220 GOTO 5000: REM  EIN PAAR
3491 :
3492 REM  * ALLGEMEINER CODE FUEER DIE KARTENWAHL *
3493 REM  EINGANG: QP$ SOFORT
3494 REM  QH HTABLE FUEER SOFORTIGE ANTWORT
3495 REM  AUSGANG: J KARTENZAHLEN
3496 REM  0 ESC
3497 REM  PROZEDUR LIESST NUR DIE WAHL NOCH NICHT BESTIMMTER PAARE ZU
3498 REM  ERLAUBT NICHT DIE WAHL VON P1
3499 :
3500 VTABLE 23: HTABLE QH: PRINT QP$:
3510 YW = 1: GOSUB 10000: REM  EINGABEN EINES ZEICHENS
3520 IF Z% = - 1 THEN J = 0: RETURN : REM  ESC
3530 IF Z% < "A" OR Z% > LC$ THEN 3700: REM  BUCHSTABE IST NICHT GUELTIG
3540 J = ASC (Z%) - ASC ("A") + 1
3550 IF CD%(J) < = 0 OR J = P1 THEN 3600: REM  KARTEN WURDE SCHON GEWAEGHLT

3560 GOSUB 8200: REM  DRUCKEN KARTEN AUS
3570 RETURN
3597 :
3598 REM  * BEREITS GEWAEGHLT *
3599 :
3600 Q$ = "DIESE KARTEN WURDE BEREITS GEWAEGHLT"
3610 GOTO 3710
3697 :
3698 REM  * WAEHLE EINEN GUELTIGEN BUCHSTABEN *
3699 :
3700 Q$ = "WAEHLE EINEN BUCHSTABEN VON A BIS " + LC$
3710 P = 60
3720 GOSUB 3900
3730 GOTO 3500: REM  VERSUCHEN NOCH EINMAL
3891 :
3892 REM  * DRUCKEN HINWEIS ZEILE *

```

```

3893 :
3894 REM   EINGANG: Q$  AUSZUDRUCKENDER TEXT
3895 REM           P   PAUSE
3899 :
3900 VTAB 24: HTAB 10: PRINT Q$;
3910 NORMAL
3920 YP = P: GOSUB 11400: REM  PAUSE
3930 HTAB 10: PRINT LEFT$ (BL$,29);: REM  LEERE HINWEIS ZEILE
3940 RETURN
3997 :
3998 REM   * KARTEN PASSEN NICHT *
3999 :
4000 Q$ = "*** KEIN PAAR ***"
4010 P = 150
4020 GOSUB 3900
4100 J = P1: REM   DREHE KARTE 1 UM,BILD NACH UNTEN
4110 GOSUB 8000
4120 J = P2: REM   DREHE KARTE 2 UM,BILD NACH UNTEN
4130 GOSUB 8000
4140 GOTO 3000: REM   WAEHLE ANDERES PAAR
4997 :
4998 REM   * KARTEN PASSEN *
4999 :
5000 Q$ = "*** RICHTIG ***"
5010 FLASH
5020 P = 150
5030 GOSUB 3900
5050 J = P1: REM   ENTFERNE KARTE 1
5060 GOSUB 5900
5070 J = P2: REM   ENTFERNE KARTE 2
5080 GOSUB 5900
5100 N = N - 2: REM   ZWEI KARTEN WENIGER
5110 IF N > 0 THEN 3000: REM   UEBRIGE KARTEN
5200 GR : HOME : REM   DRUCKE DIE ORIGINAL ANORDNUNG
5210 VTAB 21: INVERSE
5220 IF N > 0 THEN HTAB 9: INVERSE : PRINT "HIER SIND ALLE KARTEN...": NORMAL

5230 IF N = 0 THEN HTAB 2: PRINT "DU HAST ALLE PAARE IN ";TR;" VERSUCHEN GEF
      UNDEN.": NORMAL
5300 FOR J = 1 TO 2 * NP: REM   DRUCKE KARTEN MIT BILD NACH OBEN
5310 GOSUB 8200
5320 NEXT
5330 GOTO 6000: REM   NOCHMAL?
5900 GOSUB 9900: REM   ENTFERNE KARTE J
5910 PRINT " ";
5920 GOSUB 8100: REM   LOESCHE KARTE
5930 CD%(J) = - CD%(J): REM   ZEICHEN WENN PAAR GEFUNDEN
5940 RETURN
5997 :
5998 REM   ** NOCH EINMAL? **
5999 :
6000 HTAB 1: VTAB 24
6010 PRINT "NOCH EINMAL SPIELEN (J OR N)? ";
6020 GOSUB 11200: REM   J/N
6030 ON Z% + 2 GOTO 6100,6000,2000,6100: REM   ESC, UNGUELTIG, J, N
6100 PRINT : PRINT
6110 PRINT "DANKE FUER'S MITSPIELEN.";
6120 END
7997 :
7998 REM   * DRUCKE KARTEN PROZEDUREN *

```

```

7999 :
8000 COLOR= 1: REM   * DRUCKE KARTE J BILD NACH UNTEN
8010 GOTO 8110
8100 COLOR= 0: REM   * LOESCHE KARTE J
8110 H = H%(J)
8120 V = V%(J)
8130 FOR Z = H TO H + CH
8140 VLIN V,V + CV AT Z
8150 NEXT
8160 RETURN
8200 Z% = ABS (CD%(J)): REM   DRUCKE KARTE J BILD NACH OBEN
8210 REM   AENDERE ZEICHEN PROZEDUR WENN CH,CV VERAENDERT
8220 REM   AUSGANG: Z% PAARNUMMER VON KARTE J
8230 H = H%(J): REM   H-POS
8240 V = V%(J): REM   V-POS
8250 C1 = CP%(PR%(Z%,1),0): REM   FARBE 1
8260 C2 = CP%(PR%(Z%,1),1): REM   FARBE 2
8270 ON PR%(Z%,0) GOTO 8300,8400,8500,8600: REM   DRUCKE KARTEN MOTIV
8300 FOR Z = H TO H + 1: REM   MOTIV 1 = 3 V-STREIFEN
8310 COLOR= C1
8320 VLIN V,V + CV AT Z
8330 VLIN V,V + CV AT Z + 3
8340 NEXT
8350 COLOR= C2
8360 VLIN V,V + CV AT H + 2
8370 RETURN
8400 FOR Z = V TO V + 1: REM   MOTIV 2 = 3 H-STREIFEN
8410 COLOR= C1
8420 HLIN H,H + CH AT Z
8430 HLIN H,H + CH AT Z + 5
8440 COLOR= C2
8450 HLIN H,H + CH AT Z + 2
8460 HLIN H,H + CH AT Z + 3
8470 NEXT
8480 RETURN
8500 FOR Z = H TO H + 2 STEP 2: REM   MOTIV 3 = 5 V-STREIFEN
8510 COLOR= C1
8520 VLIN V,V + CV AT Z
8530 VLIN V,V + CV AT Z + 2
8540 COLOR= C2
8550 VLIN V,V + CV AT Z + 1
8560 NEXT
8570 RETURN
8600 FOR Z = V TO V + 3 STEP 3: REM   MOTIV 4 = 5 H-STREIFEN
8610 COLOR= C1
8620 HLIN H,H + CH AT Z
8630 HLIN H,H + CH AT Z + 3
8640 COLOR= C2
8650 HLIN H,H + CH AT Z + 1
8660 HLIN H,H + CH AT Z + 2
8670 NEXT
8680 RETURN
8997 :
8998 REM   ** INFORMATIONEN **
8999 :
9000 TEXT : HOME
9010 PRINT "KONZENTRATION IST EIN MEMORY-SPIEL."
9030 PRINT "DIE KARTENPAARE WERDEN GEMISCHT UND "
9040 PRINT "UMGEDREHT. VERSUCHE DIE PAARE ZU FINDEN."
9060 PRINT "DIE KARTEN SIND ENTSPRECHEND DIESEM"
9070 PRINT "DIAGRAM ANGEORDNET:"

```

```

9080 INVERSE :Z = 17
9090 HTAB Z: PRINT "ABCDEF"
9100 HTAB Z: PRINT "GHIJKL"
9110 REM HTAB Z:PRINT "MNOPQR"
9120 REM HTAB Z:PRINT "STUVWX"
9130 NORMAL
9150 PRINT "WAEHLE EINE KARTE DURCH EINTIPPEN EINES"
9160 PRINT "BUCHSTABEN VON A BIS ";LC$;". (WENN "
9170 PRINT "SIE DIE LINKE OBERE KARTEN WAEHLEN, DANN TIPPEN SIE A.)
9190 PRINT "DU KANNST NUR ZWEI KARTEN GLEICHZEITIG SEHEN. WENN DU EIN PAAR G
EFUNDEN HAST"
9200 PRINT "VERSCHWINDEN DIE ZWEI SPIELKARTEN. "
9220 PRINT "ES GIBT EINE BEGRENZUNG IN DER ZAHL DER VERSUCHE."
9240 PRINT "DRUECKE ";; INVERSE : PRINT "ESC";: NORMAL : PRINT " FUEER PROGRAM
MENDE."
9260 PRINT "MIT RETURN GEHT ES WEITER.. ";
9270 GOSUB 11500: REM WARTE AUF TASTENDRUCK
9280 IF Z% = - 1 THEN 6100: REM ESC
9290 RETURN
9497 :
9498 REM ** GEMISCHTER LOWRES-TEXT-BILDSCHIRM
9499 :
9500 GR : HOME
9510 FOR J = 1 TO 2 * NP
9520 GOSUB 8000: REM DISPLAY CARD FACE DOWN
9530 NEXT
9540 VTAB 21: HTAB 10: PRINT "JEDES ZEICHEN STEHT FUEER EINE KARTEN.
SUCHE DIE PAARE!"
9560 INVERSE
9570 FOR J = 1 TO 2 * NP
9580 GOSUB 9900
9590 PRINT CHR$ ( ASC ("A") - 1 + J);
9600 NEXT
9610 NORMAL
9620 RETURN
9900 Z% = (J - 1) / 6: REM LOKALISIERE KARTE J IN BUCHSTABEN-MATRIX
9910 VTAB 21 + Z%
9920 HTAB J - 1 - 6 * Z% + 2
9930 RETURN
49997 :
49998 REM ** VARIABLEN-VEREINBARUNGEN **
49999 :
50000 REM PR%(J,0) KARTENMOTIV VON PAAR J
50010 REM PR%(J,1) FARBENPAAR VON KARTE J
50020 REM CP%(K,0) FARBE 1 VON FARBENPAAR K
50030 REM CP%(K,1) FARBE 2 VON FARBENPAAR K
50040 REM CD%(J) PAARNUMMER VON J-TER KARTE WENN NICHT GEFUNDEN
50050 REM - PAARNUMMER WENN BEREITS GEFUNDEN
50060 REM HZ%(J) HORIZONTALE POSITION VON KARTE J
50070 REM VZ%(J) VERTIKAL
50080 REM NT ZAHL DER AKTUELLEN KARTENMOTIVE
50090 REM NC ZAHL DER AKTUELLEN KARTENPAARE
50100 REM NP ORIGINALZAHL DER PAARE
50110 REM LC$ BUCHSTABE DER LETZTEN KARTE
50120 REM N ZAHL DER NOCH NICHT GEFUNDENEN KARTEN
50130 REM CH HORIZONTALE BREITE EINER KARTE
50140 REM CV VERTIKALE HOEHE EINER KARTE
50150 REM TR ZAHL DER VERTAUSCHUNGEN
50160 REM P1 ERSTE WAHL
50170 REM P2 ZWEITE WAHL
60000 :
60010 REM * COPYRIGHT 1981 BY HOWARD FRANKLIN, PALO ALTO, CA *
60020 :

```

Geben Sie dieses Programm ein. Verbinden Sie es mit EINGABE EINHEIT und BILD EINHEIT. Speichern Sie das Programm mit dem Namen KONZENTRATION ab.

Neben der erwarteten Fehlerabfrage-Prozedur zur Vermeidung falscher Eingaben vereinigt KONZENTRATION noch andere Vorteile in sich, durch die das Programm einfach zu handhaben ist. Die Spielkarten werden von einer Matrix am linken unteren Bildschirmrand präsentiert. Diese Matrix bleibt während des ganzen Spiels auf dem Bildschirm sichtbar. Sie wird ständig erneuert, wenn Karten ausgewählt und zusammengehörige Paare gefunden werden. Auf diese Art wird der Spieler daran erinnert, nicht den gleichen Buchstaben für beide Karten eines Paares zu wählen bzw. eine Karte zu wählen, die schon mit einer anderen verbunden wurde. (Im letzten Fall verschwinden die Spielkarte und der zugehörige Buchstabe vom Bildschirm).

Frage 1: Welche Antwort druckt das Programm aus, wenn eine bereits entfernte Spielkarte gewählt wird?

Weitere Vorteile der Buchstabenmatrix sind: Der Spieler braucht keinen Spielregler zu benutzen oder sich eine komplizierte Reihenfolge von Cursorbewegungen zu merken. Alle notwendigen Informationen bleiben während des ganzen Spiels auf dem Bildschirm sichtbar. Macht der Spieler einen Fehler, druckt der Computer hilfreiche Hinweise aus. Personen, die das Spiel nicht zu Ende spielen wollen, drücken ESC und anschließend RETURN.

Um ein ansprechendes Bild zu gestalten, sollten inverse und blinkende Textdarstellungen sparsam benutzt werden. In KONZENTRATION wird die inverse Darstellung benutzt für die Nachricht "DU HAST ALLE PAARE GEFUNDEN". Die FLASH-Funktion wird benutzt, um dem Spieler ein gefundenes Paar anzuzeigen.

Frage 2: Welche Zeilennummer beinhaltet die Nachricht, daß ein Kartenpaar gefunden wurde?

Für den Spieler hat das Spiel nur einen einzigen Schwierigkeitsgrad. Wir können aber mehrere Parameter des Programms verändern und damit den Schwierigkeitsgrad steigern. Die wichtigen Ausdrücke, die die grafische Darstellung des Spiels bestimmen, sind alle als Variablen geschrieben und am Anfang des Programms festgelegt. Wir haben KONZENTRATION in einer Form entworfen, die es uns leicht macht, Zahl und Typ der dargestellten Farbmuster zu verändern. Die Farbmuster sind Kombinationen von drei oder fünf horizontalen bzw. vertikalen Streifen (siehe Zeile 1040 und 1050).

Folgende Programmänderungen sind möglich: weniger Farben (Zeile 1510), weniger horizontale und vertikale Streifen (Zeile 1500) und weniger bzw. mehr Spielkarten (Zeile 1520).

Die Spielkarten werden vor Beginn eines jeden Spiels neu gemischt. Wir haben in unserer Version zwölf Karten (sechs Paare) gewählt; die Kartenpaare suchen wir aus allen möglichen Farben und Mustern aus. Es soll zwei volle Reihen von Spielkarten in farbiger, reizvoller Anordnung geben. Sie können

aber die Parameter nach eigenem Entschluß wählen. Dabei ist darauf zu achten, daß die Anzahl der Farbstreifenkombinationen multipliziert mit der Anzahl der Farbpaare größer (oder gleich) der Anzahl der Kartenpaare ist.

Frage 3: Wie wird die Anzahl der Spielkarten am Beginn des Spieles verändert?

Das Spiel ist einfacher zu spielen, wenn die Zahl der Kartenpaare, die Zahl der Kartentypen und Farbpaare und die Zahl der Streifen auf einer Karte vermindert werden.

KONZENTRATION verfügt über keine Klangmöglichkeiten. Wir glauben, daß dieses Spiel rein visueller Natur sein sollte. Wenn Sie Klänge und Töne hinzufügen wollen, dann erweitern Sie das Programm mit KLANG EINHEIT. Geben Sie den Eingangsvariablen entsprechende Anfangswerte. Das Spiel verwendet noch viele weitere elegante Programmiertechniken. Schauen Sie sich das Listing sorgfältig an, und Sie werden einige davon entdecken.

Sterne

STERNE ist ein Ratespiel mit Zahlen, das unter dem Namen STARS am People's Community Center entwickelt wurde. Im Gegensatz zu anderen Zahlenspielen, die ebenso gut mit Bleistift und Papier zu spielen sind, nutzt STERNE die große Rechengeschwindigkeit des Computers aus. Das Programm antwortet auf jeden Rateversuch mit dem Ausdruck von Sternen anstatt irgendwelcher Kommentare. Je mehr Sterne erscheinen, desto mehr nähern wir uns dem richtigen Ergebnis. Das Programm errechnet die Anzahl der ausgedruckten Sterne, und so erhalten wir bei jedem Versuch weitere Informationen über die richtige Lösung. STARS, ursprünglich für Fernschreiber geschrieben, wurde von uns in einer LO-RES-Version für den APPLE umgeschrieben. Das Programm nutzt die Möglichkeiten der Bildschirmgestaltung des APPLE. Auch macht es von der Möglichkeit der Fehlerbearbeitung in EINGABE EINHEIT Gebrauch. Da wir in diesem Programm das ‚Pause- oder Tastendruck‘-Unterprogramm verwenden, legt es nach jeder Antwort eine Pause ein, die der Spieler durch Drücken einer Taste beenden kann.

```
10 REM    ...STERNE - EINGABE KLANG EINHEIT...
11 :
997 :
998 REM    ** INITIALISIERUNG **
999 :
1000 BP$ = CHR$ (7)
1010 L2 = LOG (2)
1020 MN = 1
1030 MX = 40
1040 S1% = LOG (MX - MN) / LOG (2) + 1
1050 TB = 13
1060 GB = 38
1070 GT = GB - 27
1197 :
```



```

1198 REM    * TITELBILD *
1199 :
1200 GR : HOME
1210 FOR Z = 1 TO 100: REM    FARBPUNKTE
1220 COLOR= 15: IF RND (1) < .75 THEN COLOR= 13
1230 PLOT INT (40 * RND (1)), INT (40 * RND (1))
1240 NEXT
1250 COLOR= 15: FOR Z = 15 TO 23: REM    WEISSES RECHTECK
1260 HLIN 4,34 AT Z
1270 NEXT
1280 X$ = "STERNE":XV = 16:XC(1) = 0: GOSUB 15300
1290 VTAB 23: HTAB 8: PRINT "MIT RETURN GEHT ES WEITER..";
1300 GOSUB 11500: REM    WARTEN AUF TASTENDRUCK
1310 IF Z% = - 1 THEN END : REM    ESC
1900 GOSUB 9000: REM    INFORMATIONEN
1997 :
1998 REM    ** INITIALISIERUNG FUER NEUES SPIEL **
1999 :
2000 GOSUB 9500: REM    GEMISCHTE TEXT-GRAFIK FORM
2010 A = INT ((MX - MN + 1) * RND (1)) + MN
2020 N = 0
2030 YL = MN
2040 YH = MX
2997 :
2998 REM    ** NAECHSTER VERSUCH **
2999 :
3000 PRINT "RATE: ";
3010 YW = 3: GOSUB 11000: REM    EINGABE INTEGER
3020 IF Z% = - 1 THEN 6200: REM    ESC
3030 IF Z% < > 1 THEN HTAB TB: PRINT "EINE ZAHL VON ";MN;" TO ";MX;" , BITTE
      ": GOTO 3000: REM    UNGUELFIGE ZAHL
3040 G = Z
3100 N = N + 1: REM    * GUELFIGER VERSUCH *
3110 IF A = G THEN 5000
3997 :
3998 REM    ** FALSCH GERATEN **
3999 :
4000 S% = S1% - INT ( LOG ( ABS ( G - A ) ) / L2)
4010 GOSUB 8000
4020 GOTO 3000
4997 :
4998 REM    ** RICHTIG GERATEN **
4999 :
5000 S% = 20
5010 GOSUB 8000: REM    BALKEN
5020 WD = 100:Z$ = "AAHHJJH": GOSUB 13300: REM    KLANG
5030 PRINT
5040 FLASH
5050 HTAB 9: PRINT "DU HAST ES IN ";N;" VERSUCH";
5060 IF N > 1 THEN PRINT "E";
5070 PRINT " GESCHAFFT!"
5080 NORMAL
5997 :
5998 REM    ** NOCHMAL? **
5999 :
6000 HTAB 1: VTAB 24
6010 PRINT "NOCH EINMAL (J OR N)? ";
6020 GOSUB 11200: REM    J/N
6030 ON Z% + 2 GOTO 6100,6000,2000,6100: REM    ESC, UNGUELFIG, J, N
6100 PRINT : PRINT

```

```

6110 PRINT "DANKE FUER'S MITSPIELEN.";
6120 END
6197 :
6198 REM  ** AUFGABE **
6199 :
6200 HTAB TB
6210 PRINT "MEINE ZAHL WAR ";A
6220 GOTO 6000
7997 :
7998 REM  ** ANTWORTE AUF VERSUCH **
7999 :
8000 HTAB TB
8010 SPEED= 120
8020 FOR J = 1 TO S%
8030 PRINT "*";BP$;
8040 NEXT
8050 SPEED= 255
8060 PRINT
8070 IF GB - 3 * S% < GT THEN S% = S% - 1: GOTO 8070
8100 REM  * DRUCKE BALKEN *
8110 COLOR= S%
8120 VLIN GB,GB - 3 * S% AT G - MN
8130 RETURN
8997 :
8998 REM  ** INFORMATIONEN **
8999 :
9000 TEXT : HOME
9010 VTAB 4
9020 HTAB 14: PRINT "*** STERNE ***"
9030 PRINT
9040 PRINT
9050 PRINT "ICH DENKE AN EINE ZAHL ZWISCHEN"
9060 PRINT MN;" UND ";MX;". VERSUCHE DIE ZAHL ZU          ERRATEN."
9070 PRINT
9080 PRINT "NACH JEDEM VERSUCH DRUCKE ICH EIN ODER "
9090 PRINT "MEHRERE STERNE (*) AUS. JE NAEHER DU"
9100 PRINT "MEINER ZAHL KOMMST, DESTO MEHR STERNE  DRUCKE ICH AUS."
9110 VTAB 20
9120 PRINT "DRUECKE ";; INVERSE : PRINT "ESC";; NORMAL : PRINT " FUER SPIELEN
    DE."
9130 PRINT
9140 PRINT "MIR RETURN GEHT ES WEITER.. ";
9150 GOSUB 11500: REM  WARTEN AUF TASTENDRUCK
9160 IF Z% = - 1 THEN 6100: REM  ESC
9200 GR : HOME
9210 X$ = "STERNE":XV = 0:XC(1) = 13: GOSUB 15300
9220 PRINT "....5...10...15...20...25...30...35...40"
9230 POKE 34,22: REM  SETZE TEXTFENSTER
9290 RETURN
9497 :
9498 REM  ** GEMISCHTE LOWRES-GRAFIK GESETZT **
9499 :
9500 COLOR= 0: FOR Z = GT TO GB: HLIN 0,39 AT Z: NEXT : REM  LOESCHE GRAFIKB
    EREICH
9510 HOME : REM  LOESCHE TEXTFENSTER
9520 RETURN
49991 :
49992 REM  *** STERNE ***
49993 :
49994 REM  ORIGINALVERSION DER PEOPLE'S COMPUTER COMPANY, MENLO PARK, CA
49997 :

```

```

49998 REM  ** VARIABLENVEREINBARUNGEN **
49999 :
50000 REM  A  ANTWORT
50010 REM  BP$ BEEP (CHR$(7))
50020 REM  F  FLAG FUER GUELTIGE EINGABE
50030 REM  G  RATEVERSUCH
50040 REM  GB  BODEN DES GRAPHEN
50050 REM  GT  KOPF DES GRAPHEN
50060 REM  J  SCHLEIFEN ZAEHLER
50070 REM  L2  LOG(2)
50080 REM  MN  MINIMALE ANTWORT
50090 REM  MX  MAXIMUM
50100 REM  N  ZAHL DER VERSUCHE
50110 REM  S%  ZAHL DER STERNE FUER VERSUCH
50120 REM  S1% MAXIMUN DER STERNE +1
50130 REM  TB  TAB POSITION FUER ANTWORT
60000 :
60010 REM  * COPYRIGHT 1981 BY HOWARD FRANKLIN, PALO ALTO, CA *
60020 :

```

Verbinden Sie dieses Programm mit EINGABE EINHEIT, KLANG EINHEIT (löschen Sie die Zeilen 18000 bis 19999) und BILD EINHEIT. Speichern Sie es unter dem Namen STERNE.

Wie wir wissen, steht uns bei der LO-RES-Grafik nur ein vier Zeilen großes Textfenster am unteren Bildschirmrand zur Verfügung. Deshalb haben wir die Informationen zu diesem Spiel an den Anfang des Programms gesetzt. Durch die optische Wiedergabe auf dem Bildschirm bleibt die Idee des Spiels deutlich erkennbar.

In diesem Spiel werden alle Hinweise eindrucksvoll auf dem Bildschirm ausgedruckt. Die Darstellung der Zahlen in einer Reihe ermöglicht eine visuelle Anordnung der Information, die in den früheren Fernschreiber-Versionen nicht möglich gewesen ist. Alle Hilfen und Hinweise bleiben auf dem Monitor ständig sichtbar. Deshalb hielten wir es für vertretbar, stets nur eine Antwortzeile im Textfenster des Bildschirms erscheinen zu lassen.

Da wir die Grafik nicht nur als Dekoration verwenden, sondern in das Spiel integrieren, ist diese Version von STERNE ein sehr hübsches Spiel. Wenn Sie Programme entwickeln bzw. verbessern, überlegen Sie, wie grafische Effekte im Programm verwendet werden können. Zum Beispiel können Linien und Bilder dazu benutzt werden, hilfreiche Informationen zu vermitteln.

Wir haben die Zahl der erlaubten Rateversuche nicht begrenzt, wie das sonst in diesen Spielen üblich ist. Das Einschränken der erlaubten Versuche kann Kinder vom Spielen fernhalten. Durch Verwendung unserer ESC-Vereinbarung kann der Benutzer wählen, ob er weitermachen oder aufgeben will. Das ist weitaus angenehmer und spielerfreundlicher.

Simon

SIMON ist unsere Version des populären Spiels, in dem der Computer eine Melodie spielt, die der Spieler ohne Fehler nachspielen soll.

```
10 REM    ...SIMON-EINGABE KLANG
11 :
997 :
998 REM  ** INITIALISIERUNG **
999 :
1000 DIM T%(30)
1100 Q1 = 1: REM    WIEDERHOLE ZAHL IN MELODIE OPTION
1110 Q2 = 1: REM    NOTE UNTER BOX OPTION
1120 Q3 = 1: REM    WAEHLE BOX OPTION
1130 Q4 = 1: REM    KLANG NOTE OPTION
1200 TL = 1: REM    TIEFSTE NOTE
1210 TH = 8: REM    HOECHSTE NOTE
1220 TD = 80: REM    LAENGE JEDER NOTE
1230 TP = 10: REM    PAUSE ZWISCHEN NOTEN
1240 LL = 3: REM    MINIMALE LAENGE JEDER MELODIE
1250 LH = 20: REM    MAXIMUM
1300 BV = 3: REM    BOX HOEHE
1310 VR = 30: REM    V-POS DER BOX
1320 VS = VR - BV - 8: REM    V-POS DER GEWAEHLTEN BOX
1330 HT = 19: REM    H-POS DER MELODIE
1340 VC = 23: REM    V-POS DER COMPUTER MELODIE
1350 VL = 24: REM    V-POS DER LAENGE
1360 VY = 22: REM    V-POS IHRER MELODIE
1400 CB = 1: REM    HINTERGRUND FARBE
1410 CR = 2: REM    BOX FARBE
1420 CR = 3: REM    FARBE GEWAEHLTER BOX
1500 L = LL: REM    LAENGE DES ERSTEN TONES
1597 :
1598 REM  * TITELBILD *
1599 :
1600 TEXT : HOME
1610 WP = 0: GOSUB 13000: REM    VERMEIDE INITIALISIERUNGS VERZOEGERUNG BEI ERS
    TER NOTE
1620 S$ = "SIMON"
1630 FOR J = 1 TO LEN (S$)
1640 VTAB 16: HTAB 17 + J: INVERSE : PRINT MID$ (S$,J,1)
1650 VTAB 12 - J: HTAB 4 * J: PRINT MID$ (S$,J,1);: NORMAL
1660 WP = J:WD = 60: GOSUB 13000: REM    NOTE J
1670 YP = 15: GOSUB 11400:
1680 PRINT CHR$ (8);" "
1690 NEXT
1700 SPEED= 100: FOR Z = 1 TO 4: PRINT CHR$ (7);: NEXT : SPEED= 255
1710 VTAB 23: HTAB 8: PRINT "MIT RETURN GEHT ES WEITER..";
1720 GOSUB 11500: REM    WARTEN AUF TASTENDRUCK
1730 IF Z% = - 1 THEN END : REM    ESC
1900 REM    GOSUB 9000: REM    INFORMATIONEN
1997 :
1998 REM  ** INITIALISIERUNG FUER NAECHSTE NOTE **
1999 :
2000 FOR J = 1 TO L: REM    MACHE MELODIE
2010 T%(J) = (TH - TL) * RND (1) + TL
2020 NEXT
2030 GOSUB 8300: REM    SETZE BOXEN
2040 VT = VC:Q9 = Q1:Q1 = 0: GOSUB 8900: REM    SPIELE COMPUTER MELODIE(DRUCKE
    NICHT MELODIE)
```

```

2050 Q1 = Q9: REM   SPEICHERE MELODIE OPTION
2060 YP = 10: GOSUB 11400: REM   PAUSE
2100 HOME
2110 VTAB VL: HTAB HT - 8: PRINT "LAENGE: ";L;: REM   DRUCKE LAENGE
2120 VTAB VY: HTAB HT - 11: PRINT "DEINE MELODIE";
2130 NJ = 1: REM   ERSTE NOTE
2050 Q1 = Q9: REM   SPEICHERE MELODIE OPTION
2060 YP = 10: GOSUB 11400: REM   PAUSE
2100 HOME
2110 VTAB VL: HTAB HT - 8: PRINT "LAENGE: ";L;: REM   DRUCKE LAENGE
2120 VTAB VY: HTAB HT - 11: PRINT "DEINE MELODIE";
2130 NJ = 1: REM   ERSTE NOTE
2997 :
2998 REM   ** NAECHSTE NOTE IN MELODIE **
2999 :
3000 GOSUB 11600: REM   TASTENDRUCK, OHNE WIEDERGABE, OHNE VORHERIGES ZEICHEN
3010 IF Z% = - 1 THEN VTAB VY: HTAB HT + 2 + NJ: FLASH : PRINT "ESC";: NORMAL : GOTO
      4000: REM   ESC
3020 N = Z - 176: REM   WANDLE UM IN NOTE #
3030 IF N < TL OR N > TH THEN 3000: REM   UNGUELTIGE NOTE
3040 VT = VY: GOSUB 8600: REM   ANTWORTE AUF NOTE
3050 IF N = T%(NJ) THEN 5000: REM   RICHTIGE NOTE
3997 :
3998 REM   ** MELODIE WAR FALSCH **
3999 :
4000 SPEED= 100: FOR Z = 1 TO 4: PRINT CHR$( 7);: NEXT : SPEED= 255
4010 YP = 10: GOSUB 11400: REM   PAUSE
4020 Q9 = Q1:Q1 = 1: GOSUB 8800: REM   SPIELE COMPUTERMELODIE(DRUCKE MELODIE)
4030 Q1 = Q9: REM   SPEICHERE PRINT NOTE OPTION
4040 IF L > LH THEN L = L - 1
4050 GOTO 6000: REM   NOCHMAL?
4997 :
4998 REM   ** RICHTIGE NOTE **
4999 :
5000 NJ = NJ + 1
5010 IF NJ < = L THEN 3000: REM   NAECHSTE NOTE
5097 :
5098 REM   ** RICHTIG **
5099 :
5100 HOME
5110 VTAB VY: HTAB HT - 12: PRINT "DAS IST RICHTIG:";
5120 Q9 = Q1:Q1 = 1: GOSUB 8900: REM   SPIEL DIE MELODIE
5130 Q1 = Q9: REM   SPEICHERE PRINT MELODIE OPTION
5140 REM   ** NAECHSTE MELODIE IST LAENGER **
5150 IF L < LH THEN L = L + 1
5997 :
5998 REM   ** NOCHMAL? **
5999 :
6000 HTAB 1: VTAB 24
6010 PRINT "NAECHSTE MELODIE (J OR N)? ";
6020 GOSUB 11200: REM   J/N
6030 ON Z% + 2 GOTO 6100,6000,2000,6100: REM   ESC, UNGUELTIG, J, N
6100 PRINT : PRINT
6110 PRINT "VIELEN DANK FUER'S MITSPIELEN.";
6120 END
7997 :
7998 REM   ** DRUCKE BOX UNTERPROGRAMME **
7999 :
8000 COLOR= CR: REM   * DRUCKE BOX N
8010 Z = VR
8020 GOTO 8200

```

```

8050 COLOR= CB: REM   * LOESCHE BOX N
8060 GOTO 8010
8100 COLOR= CS: REM   * DRUCKE GEWAEHLTE BOX N
8110 Z = VS
8120 GOTO 8200
8150 COLOR= CB: REM   * LOESCHE GEWAEHLTE BOX N
8160 GOTO 8110
8200 Z1 = - 4 + 5 * N: REM   * DRUCKE BOX N IN REIHE Z
8210 REM   N = BOX #
8220 REM   FARBE FUER DRUCKEN ODER LOESCHEN
8230 REM   Z = ROW #
8240 FOR Z2 = Z + BV - 1 TO Z STEP - 1: REM   DRUCKE VON BODEN
8250 HLIN Z1,Z1 + 2 AT Z2
8260 NEXT
8270 RETURN
8300 GR : HOME : REM   * DRUCKE HINTERGRUND UND ERSTE BOXEN
8310 COLOR= CB
8320 FOR Z = 0 TO 39
8330 HLIN 0,39 AT Z
8340 NEXT
8350 FOR N = TL TO TH
8360 GOSUB 8000
8370 NEXT
8380 RETURN
8400 Z$ = STR$(N): REM   * WIEDERHOLE ZAHL UNTER BOX N
8410 INVERSE
8420 GOTO 8500
8450 Z$ = " ": REM   * LOESCHE ZAHL UNTER BOX N
8500 VTAB 21: REM   * DRUCKE Z$ UNTER BOX N
8510 HTAB - 2 + 5 * N
8520 PRINT Z$;
8530 NORMAL
8540 RETURN
8600 REM   * ANTWORT AUF NOTE N,NJ NOTE IN MELODIE, WIEDERHOLE NOTE IN ZEILE VT
8610 IF Q1 > 0 THEN VTAB VT: HTAB HT + 3 + NJ: INVERSE : PRINT N$: NORMAL : REM
WIEDERHOLE NOTE IN MELODIE
8620 IF Q2 > 0 THEN GOSUB 8400: REM   WIEDERHOLE NOTE UNTER BOX
8630 IF Q3 > 0 THEN GOSUB 8050: GOSUB 8100: REM   WAEHLE BOX
8640 IF Q4 > 0 THEN WD = TD:WP = N: GOSUB 13000: REM   SPIELE NOTE
8650 YP = TP: GOSUB 11400: REM   PAUSE
8660 IF Q3 > 0 THEN GOSUB 8150: GOSUB 8000: REM   WAEHLE BOX
8670 IF Q2 > 0 THEN GOSUB 8450: REM   LOESCHE NOTE UNTER BOX
8680 RETURN
8800 VT = VC: REM   * SPIELE DIE COMPUTER MELODIE
8810 IF Q1 > 0 THEN VTAB VT: HTAB HT - 17: PRINT "COMPUTER MELODIE:";
8900 REM   * SPIELE MELODIE IN ZEILE VT *
8910 FOR NJ = 1 TO L
8920 N = I%(NJ)
8930 GOSUB 8600: REM   ANTWORTE AUF NOTE NJ IN MELODIE
8940 NEXT
8950 RETURN
8997 :
8998 REM   ** INFORMATIONEN **
8999 :
9000 TEXT : HOME
9200 PRINT
9210 PRINT "DRUECKE ";; INVERSE : PRINT "ESC";: NORMAL : PRINT " FUER SPIELLENDE ,"
9220 PRINT
9230 PRINT "MIT RETURN GEHT ES WEITER.. ";
9240 GOSUB 11500

```

```

9250 IF ZZ = - 1 THEN 6100: REM ESC
9260 RETURN
13117 :
13118 REM PITCHES
13119 :
13120 DATA 255,228,203,192,171,152,135,127
60000 :
60010 REM * COPYRIGHT 1981 BY HOWARD FRANKLIN, PALO ALTO, CA *
60020 :

```

Verbinden Sie dieses Programm mit EINGABE EINHEIT und KLANG EINHEIT. Speichern Sie es als SIMON.

Die Aufgabe des Spielers ist es, die Melodie des Computers zu wiederholen. Dabei benutzt er die Zifferntasten. Unser Spiel gibt sowohl sichtbare als auch hörbare Hilfen. Die farbigen Kästchen bewegen sich, und die zu den Tönen gehörigen Zahlen erscheinen auf dem Bildschirm. Der Spieler kann sich auf die Zahlen, auf die relative Position der Kästchen, auf die Töne oder auf eine Kombination dieser drei Möglichkeiten konzentrieren.

Schauen Sie sich die Variablen an, mit denen das Spiel geändert werden kann (Zeile 1100 bis 1500). Die Änderung der Variablen ergibt eine enorme Vielfalt von möglichen Variationen. Damit das Monitorbild nicht zu verwirrend wird, haben wir allen Kästchen dieselbe Farbe zugeordnet. Die Farbe der Kästchen und die des Hintergrundes können in den Zeilen 1400 bis 1420 verändert werden. Jeder Ton der Melodie unterscheidet sich vom vorhergehenden Ton. Er wird per Zufallsfunktion aus den vorhandenen Tönen ausgewählt. Die Schwierigkeit des Spiels ergibt sich allein aus der Länge der Melodien. Lange Melodien sind schwer, kurze Melodien leicht nachzuspielen. Der Erfolg des Spielers mit einer Melodie bestimmt, ob die nachfolgende Melodie leichter oder schwerer sein wird: Das Programm paßt sich von selbst den Fähigkeiten des Spielers an. Wir können das Spiel erschweren, indem wir die Pause zwischen den einzelnen Tönen verkürzen.

Frage 4: Welche Zeile muß dazu geändert werden?

Die Zahl der möglichen Töne (und der Kästchen) wird im Programm festgelegt. Wir haben acht Töne benutzt.

Frage 5: Wie wird das Programm geändert, damit nur noch fünf Töne verwendet werden?

Frage 6: Die Dauer des ersten zu spielenden Tones soll fünf Einheiten betragen. Welche Zeile wird geändert?

Frage 7: Wie wird das Programm geändert, damit die Zahlen, die beim Spielen jeder Note erscheinen, verschwinden?

Das Programm reagiert sofort auf das Betätigen der ersten falschen Taste. Ist zum Beispiel eine Melodienfolge 3, 5, 4, aber der Spieler betätigt die Tasten 3, 6, 4, dann stoppt das Programm bei 6. Es zeigt an, daß die Melodie falsch war, und wiederholt die richtige Tonfolge.

Zusammenfassung von Kapitel 7

In diesem Kapitel haben wir drei besonders hübsche Spiele vorgestellt. STERNE ist eine technisch moderne Version eines alten Computerspiels. KONZENTRATION und SIMON sind moderne Computerversionen populärer Spiele, die wir dem Computer-Zeitalter angepaßt haben. Zusammen mit den vielen Variationsmöglichkeiten bekommen wir eine Vielzahl von unterschiedlichen KONZENTRATION- und SIMON-Spielen. Wir wünschen dem Leser viel Freude damit!

Lösungen zu Kapitel 7

Frage 1: DIESE KARTE WURDE BEREITS GEWAEHLT
(siehe die Programmzeilen 3598 bis 3610.)

Frage 2: Zeile 5000. Beachten Sie, daß Q\$ auch für den Ausdruck andere Hinweise in den Zeilen 3600, 3700 und 4000 verwendet wird. Außerdem wird Q\$ jedesmal in Zeile 3900 ausgedruckt.

Frage 3: 1520 NP = 12 (Die maximale Anzahl von Farbenpaaren ist 12 – siehe Zeile 1010.)

Frage 4: 1230 TP =

Frage 5: 1210 TH = 5 (Oder ändern Sie TL und TH so, daß zwischen diesen zwei Werten fünf Noten liegen.)

Frage 6: 1240 LL = 5

Frage 7: 1110 Q2 = 0

Anhang A

RENUMBER/MERGE-Hilfsprogramm

Zum leichteren Gebrauch der vorgestellten Programme und Unterprogramme werden sie mit den eigenen Programmen des Lesers verbunden. Das erfordert in einigen Fällen, daß die eigenen Programme umnummeriert werden. Auf der System Master Diskette, die zum Lieferumfang eines jeden APPLE-Computers gehört, befindet sich ein Hilfsprogramm zur Verbindung (Merge) und Umnummerierung (Renumber).

An dieser Stelle wollen wir kurz zeigen, wie das Hilfsprogramm anzuwenden ist (durch Starten des Programms RENUMBER INSTRUCTIONS erscheint auf dem Bildschirm der komplette Befehlssatz).

1. Starten Sie das RENUMBER-Programm. Es wird eingelesen und in den höheren Speicher-Bereich des Rechners abgelegt.
2. Lesen Sie Ihr eigenes Programm mit dem Befehl LOAD NAME1 RETURN ein.
3. Tippen Sie: &H RETURN. Das Programm wird nun festgehalten.
4. Lesen Sie das zweite Programm mit dem Befehl LOAD NAME 2 RETURN ein.
5. Verbinden Sie beide Programme mit: &M RETURN.

Das Ergebnis befindet sich nun im Programmspeicher des APPLE. Bevor Sie irgendetwas anderes durchführen, sollte zuerst das Programm mit einem eigenen Namen auf Diskette gespeichert werden. Das komplette Programm kann dann gestartet werden.

Wir haben unsere Programme und Unterprogramme so numeriert, daß sie sich voraussichtlich nicht mit den Programmen des Lesers überschneiden. Es ist wichtig, daß die Zeilennummern von zwei Programmen sich nicht überschneiden. Wenn sie dies doch tun, treten einige unangenehme Folgen auf. Haben zum Beispiel zwei Befehle die gleiche Zeilennummer, tritt diese Zeilennummer im resultierenden Programm zweimal auf. Zur Vermeidung solcher Probleme sollten die Programmzeilen im eigenen oder im gewünschten Unterprogramm so umnummeriert werden, daß sie sich nicht mehr überlappen. Dazu

kann wieder das RENUMBER-Programm benutzt werden. Das Verfahren wird wie folgt durchgeführt:

1. Starten Sie das RENUMBER-Programm
2. Lesen Sie das Programm ein, das umnummeriert werden soll.
3. Tippen Sie: & RETURN.

Das ganze Programm wird nun neu numeriert. Die neuen Zeilennummern beginnen mit der Zeile 10 und werden in Zehnerschritten fortgesetzt. Alle Befehle, die auf Programmzeilen Bezug nehmen, wie GOTO, GOSUB, ON ...
... GOTO und IF ... THEN werden automatisch richtiggestellt. Die Umnummerierung eines 16K Programms nimmt ungefähr eine Minute in Anspruch. Während dieser Zeit hat es den Anschein, als ob der Rechner untätig sei. Glauben Sie bitte nicht, daß ein Fehler vorliegt. Drücken Sie daher auf keinen Fall RESET. Anschließend können Sie das Programm speichern, starten oder auflisten lassen.

Der Umnummerierungs-Prozeß kann auch bei einer anderen Zeilennummer als bei 10 beginnen oder in anderen Schrittweiten erfolgen. Auch können einzelne Teile eines Programms neu numeriert werden, ohne das ganze Programm zu verändern. Dazu werden folgende Symbole benötigt:

- F — gibt die Nummer der ersten neuen Zeile an
- I — gibt die Differenz zwischen zwei aufeinanderfolgenden Zeilen an
- S — gibt die erste Zeile an, die neu numeriert werden soll
- E — gibt die letzte Zeile an, die neu numeriert werden soll.

Einige Beispiele sind:

&F100, I20, S350, E660 — Umnummerierung der Zeilen 350 bis 660 beginnend mit Zeile 100 und der Schrittweite 20. Die neuen Zeilennummern lauten 100, 120, 140, ...

&S1000, E2500, F1000, I15 — Umnummerierung der Zeilen 1000 bis 2500 beginnend mit Zeile 1000 und Schrittweite 15. Die neuen Zeilennummern lauten 1000, 1015, 1030, ...

Anhang B

Randbemerkungen zum Programmieren der Spiele

Dieser Anhang ist für den fortgeschrittenen Programmierer geschrieben und erläutert die Überlegungen, die einigen Programmen dieses Buches zugrunde liegen. Wir erläutern die Notwendigkeit einer Unterprogramm-Bibliothek und die Einschränkungen in APPLESOFT BASIC, die Konstruktion und den Gebrauch einer solchen Bibliothek betreffend. Assemblerlistings einiger Hauptprogrammteile, die wesentlich sind, aber nicht in BASIC geschrieben werden können, sind in diesem Anhang enthalten. Auch finden wir hier eine Anzahl von Bemerkungen zu den vorgestellten Programmen. Die Bemerkungen sind in technischer Hinsicht zu speziell, als daß wir sie an anderer Stelle hätten bringen können. (Manchmal werden sie auch 'Ramblings' genannt.)

Auf keinen Fall soll hier eine gründliche, jeden einzelnen Schritt beschreibende Untersuchung eines jeden Algorithmus durchgeführt werden. Die REM-Anweisungen in jedem Listing skizzieren ausführlich den Programmverlauf. Zur Beantwortung spezieller Fragen können diese REM-Anweisungen herangezogen werden.

Unterprogramm-Bibliothek

Aus der Sicht des Programmierers vergrößert ein Unterprogramm die Möglichkeiten einer gegebenen Programmiersprache. Wird ein Unterprogramm entwickelt und fertiggestellt, entspricht es einem „Super-Befehl“. Einige Unterprogramme sind spezielle „Super-Befehle“ für spezielle Anwendungen (z. B. das Ausdrucken einer veränderlichen Anzahl von Sternen im Programm STERNE, Zeile 8000). Einige andere Unterprogramme besitzen mehrere Anwendungsmöglichkeiten (z. B. Eingabe und Wiederholung eines Textes, Abfrage für ESC-Taste oder Überprüfung einer Integer-Zahl innerhalb vorgegebener Grenzen).

Eine Unterprogramm-Bibliothek ist einfach eine Sammlung solcher Unterprogramme, die allgemein verwendet werden können. In diesem Buch haben wir vier Unterprogramm-Einheiten (Zusammenfassung mehrerer Unterprogramme) entwickelt. Jede Einheit vergrößert die Leistungsfähigkeit von APPLESOFT.

EINGABE EINHEIT erweitert die Möglichkeiten der INPUT/GET-Befehle. KLANG EINHEIT führt neue Klangmöglichkeiten ein. BILD EINHEIT beeinflusst Bilder in LO-RES-Grafik, und NEXTDATA EINHEIT ermöglicht die RESTORE-Funktion bei jeder beliebigen Zeilennummer und nicht nur bei der ersten DATA-Anweisung. Für die Benutzung dieser Unterprogramme greifen Sie bitte auf die Kapitelzusammenfassungen zurück. Über die Unterprogramme werden wir später einige Zusatzbemerkungen machen.

Probleme bei der Erstellung einer Unterprogramm-Bibliothek

Bei der Erstellung einer Unterprogramm-Bibliothek sind zwei Arten von Problemen zu lösen. Die erste betrifft die Einschränkungen, die sich durch die Verwendung einer bestimmten Programmiersprache ergeben. In APPLESOFT gibt es drei davon: Variablen-Konflikte (das Ändern der Werte von Variablen im Unterprogramm, die auch im Hauptprogramm benutzt werden), Zeilennummer-Konflikte (die Zeilennummern im Haupt- und Unterprogramm überlappen sich), und DATA-Anweisungen-Konflikte (die Unfähigkeit, Daten aus einer bestimmten Programmzeile zu lesen, weil DATA-Anweisungen in anderen Unterprogrammen zuerst berücksichtigt werden). Andere Programmiersprachen, sogar andere BASIC-Versionen, schalten einige dieser syntaktischen Probleme aus. Die Einführung ‚lokaler‘ Variablen eliminiert das erste Problem. Sprachen ohne Zeilennumerierung vermeiden das zweite, und „RESTORE X“ mit X als Zeilennummer beseitigt das dritte Problem.

Die zweite Art von Problemen betrifft die Schwierigkeiten beim Gebrauch der Bibliothek. Für das Aufrufen von Teilprogrammen muß genau dokumentiert werden, welche Eintritts- und Austrittsbedingungen erforderlich sind. Initialbedingungen müssen ebenfalls angegeben werden (z.B. „Lese maschinensprachliches Programm X an der Stelle Y vor Benutzung des Unterprogramms“). Besonders wichtig ist die sorgfältige Anpassung der Unterprogramme, um unerwünschte Nebeneffekte zu vermeiden (z.B. Ausdruck von „AUSSERHALB DES ZULAESSIGEN WERTEBEREICHES“). Damit werden die Unterprogramme für viele Bereiche unbrauchbar. Alle diese Probleme sind im allgemeinen nicht von einer bestimmten Programmiersprache abhängig, sondern hängen von der sorgfältigen Planung des Programmierers ab.

Ausgewählte Lösungen

Es gibt keine absolut „richtige Antwort“ auf diese Probleme, sondern eine Vielzahl von Lösungswegen. Die in diesem Buch vorgestellten Möglichkeiten sind eine Auswahl der „besten Lösungen“, die von Programmierern aus objektiven und subjektiven Gründen erarbeitet wurden (z.B. leichteres Anpassen, aus ästhetischen Gründen, persönlichen Vorlieben).

Problem Nr. 1: Variablennamen

Variablennamen, die mit W beginnen, haben wir für KLANG EINHEIT reserviert. Die Anfangsbuchstaben weiterer Variablen sind: X für BILD EINHEIT, Y für EINGABE EINHEIT und NEXTDATA EINHEIT und Z für Zeitvariablen. Im allgemeinen sollte das Hauptprogramm nur Variablennamen benutzen, die mit A, B, C, ..., V beginnen. Diese Lösung erscheint auf den ersten Blick willkürlich, da nicht viele der möglichen Variablennamen im Bereich von W bis Z benutzt werden. Eine Alternative ist die Auswahl eines kleinen, mehrmals verwendbaren Satzes von Variablen. Die wirklich reservierten Variablennamen müssen gesondert aufgeführt werden. Dies ist jedoch aus verschiedenen Gründen keine „einfache“ bzw. „elegante“ Lösung: Es ist einfacher sich zu merken, daß die Buchstaben W bis Z nicht benutzt werden dürfen, als sich bestimmte reservierte Namen zu merken. Es ist auch schwieriger, ein Überschneiden der einzelnen Unterprogramme in Bezug auf die Variablen zu vermeiden. Auch ist das Verbinden der einzelnen Einheiten schwieriger, wenn solche willkürlichen Variablennamen benutzt werden.

BASIC-Programme sind immer schwierig zu lesen. Daher werden die Variablennamen so ausgewählt, daß sie, wenn möglich, auch als Gedächtnishilfe dienen. So bedeutet z.B. XH die horizontale Position in BILD EINHEIT und YH die größte Integer-Zahl innerhalb eines Intervalls im Programm EINGABE EINHEIT. YM kann sowohl Minimum als auch Maximum bedeuten. Unsere vorgeschlagene Lösung erzeugt also eine einfachere und ‚schönere‘ Bezeichnungsweise.

Im folgenden geben wir noch einige zusätzliche Regeln für die Wahl von Variablennamen an. Vermeiden Sie die Buchstaben I und O — sie sind zu leicht mit 1 und 0 zu verwechseln. Eine weitere Vereinbarung ist die Benutzung von Integer-Variablen für die Ausgabe von Code- oder Steuerzahlen ($Z\% = -1$ ESC; $= 0$ ungültige Integer-Zahl; $= 1$ gültige Integer-Zahl). Dies gilt nicht für die Ausgabe von Zahlenwerten. (Z = Zahlenwert, wenn Integer-Zahl gültig ist). Auch sollten Integer-Zahlen für Kennzeichnungen (flags) benutzt werden ($WR\% > 0$, wenn KLANG EINHEIT bereits eingelesen). Wenn es möglich ist, sollten zum Abspeichern von Werten möglichst Integer-Felder anstatt Real-Felder benutzt werden (z.B. $L\% ()$ und $R\% ()$ in MATCH). Benutzen Sie $INT ()$ anstatt einer Integer-Variablen. Dies ist im Programmlisting leichter zu verfolgen.

Nicht betrachtet haben wir bisher die Beschleunigung des Programmablaufs durch sorgfältiges Anordnen der Variablen (siehe auch APPLESOFT II Benutzer-Handbuch, Anhang E). Es gibt keinen einfachen Weg, diese Möglichkeit in eine Unterprogramm-Bibliothek mit aufzunehmen, aber vielleicht versuchen fortgeschrittene Programmierer, die Schwierigkeit zu beheben? Wir halten einen klaren Programmaufbau für wichtiger und haben uns entschlossen, dieses Problem zu ignorieren. Mit Ausnahme von BILD EINHEIT sind alle Unterprogramme schnell genug.

Problem Nr. 2: Zeilennumerierung

Die Zeilen 10xxx und 11xxx sind für EINGABE EINHEIT reserviert, die Zeilen 13xxx für KLANG EINHEIT, 15xxx für BILD EINHEIT und 19xxx für NEXTDATA EINHEIT. Außerdem sind die Zeilen 20100 bis 49999 für die Bilderbibliothek von BILD EINHEIT vorgesehen.

Die Lösung kann mit der Lösung für die Variablennamen verglichen werden. Eine Alternative wäre, das RENUMBER-Programm nicht nur zum Verbinden von Programmen, sondern auch zur Umnummerierung zu benutzen. Die Unterprogramme könnten einfach neu numeriert werden und damit im Listing dort platziert werden, wo gerade Platz ist. Der größte Nachteil dieser Lösung ist das Variieren der Eintrittspunkte von Programm zu Programm; variable Eintrittspunkte sind wesentlich schwieriger zu handhaben als feste. Außerdem dürften doch genügend Programmzeilen für das Hauptprogramm übrig sein. Ein übersichtlich aufgebautes Programm kann z. B. so aussehen:

- Unterprogramm-Einheiten beginnen ab Zeile 10000
- größere logisch zusammenhängende Teile beginnen mit 1000, 2000, ... usw.
- kleinere Teile beginnen mit 100, 200, ... usw.

Häufiges Umnummerieren macht es schwerer, dem Programmverlauf zu folgen (und der Aufbau wird unübersichtlich und häßlich).

GOTO- oder GOSUB-Befehle sollten nie zu Programmzeilen führen, die nur REM-Anweisungen enthalten, denn in manchen Fällen werden sie vielleicht gelöscht oder beim Eintippen ausgelassen. Ein Eintritt sollte nur am Anfang eines Unterprogramms erfolgen. Der trickreiche Eintritt in die Mitte eines Unterprogramms ist riskant, und das Programmlisting ist später schwierig abzuändern. Dies kann durch Neustrukturierung der lokalen Variablen und der Eingangsvariablen des Unterprogramms vermieden werden.

Im Bestreben, gute Lesbarkeit der Listings zu erreichen, wollten wir die Auswahl der Variablennamen, die Zeilennumerierung und die Benutzung der REM-Anweisungen in jedem Programm übereinstimmend anwenden. Deshalb sehen sich die Programmlistings alle etwas ähnlich. Auch das Empfinden des Programmierers für einen „schönen“ Programmaufbau steigerte sich während dieses Prozesses, mit dem Erfolg, daß die späteren Programme stärker übereinstimmen als die ersten. Es ist sehr schwierig, schöne und klar strukturierte Listings in BASIC zu schreiben. Unsere Programme stellen den Versuch dar, übersichtliche Listings zu entwickeln.

Ebenso wie das sorgfältige Anordnen der Variablen, kann auch das überlegte Anordnen der Zeilennummern die Ausführung eines Programms beschleunigen (siehe wieder APPLESOFT II Programmieranleitung, Anhang E). Aus den gleichen Gründen, wie bereits erwähnt, ignorieren wir dieses Problem.

Problem Nr. 3: DATA-Anweisungen

Die Lösung dieses Problems erfolgt direkt und ist schwierig. Ein "RESTORE X"-Befehl (X kann eine beliebige Zeilennummer sein) wurde in NEXTDATA

EINHEIT zusätzlich aufgenommen. Viele BASIC-Varianten besitzen diesen Befehl bereits, aber unglücklicherweise nicht APPLESOFT.

Die Bilderbibliothek in BILD EINHEIT benötigt keinen großen Aufwand für die Verwaltung der Bildadressen, denn jedes Bild beginnt mit einer Zeile $20000 + 100 * \text{ZAHL}$. Sie kann daher einfach mit RESTORE X ergänzt werden.

Das Programm KLANG EINHEIT liest ein maschinensprachliches Unterprogramm durch die Verwendung von POKE-Befehlen und DATA-Anweisungen ein, statt einzelne POKE-Befehle zu verwenden. (Dagegen wird in NEXT-DATA EINHEIT das Assemblerlisting nur mit POKE-Befehlen gelesen). Hier ist ein Assemblerlisting von RESTORE X:

```

6      *
7      * APPLESOFT ANGEFASST
8      *
9      DATPTR      EQU    $7D ;MEMORY POS FUER
      NAECHSTES EINLESEN
10     LINNUM      EQU    $50 ;ZEILENNUMMER FUER 'FNDLIN'
11     LOWPTR      EQU    $9B ;ADRESSE VON 'FNDLIN'
12     FNDLIN      EQU    $D61A ;SUCHE FUER ZEILENNUMMER
13
14
15     * RESTOREX - NAECHSTES EINLESEN VON ZEILE X
16     *
17     LINEX       DS     Z ;ZEILENNUMMER
18     *
0302: AD 00 03 19     RESTOREX LDA    LINEX;SETZE ZEILENNUMMER
0305: 85 50      20             STA    ZEILENNUM
0307: AD 01 03 21             LDA    LINEX+1
030A: 85 51      22             STA    ZEILENNUM+1
030C: 20 1A D6 23             JSR    FNDLIN ;SUCHE
030F: A5 9B      24             LDA    LOWPTR ; ERNEUERE ZEIGER
      FUER NAECHSTES EINLESEN
0311: 18         25             CLC
0312: 69 04      26             ADC    #4 ; AKTUELLE DATEN
0314: 85 7D      27             STA    DATPTR
0316: A5 9C      28             LDA    LOWPTR+1
0318: 69 00      29             ADC    #0
031A: 85 7E      30             STA    DATPTR+1
031C: 60         31             RTS

```

Problem Nr. 4: Beschreibung des Aufrufs von Programmteilen

Die Kapitelzusammenfassungen beschreiben alle aufrufbaren Programmteile zu jedem Unterprogramm. Zusätzlich stehen in den Listings REM-Anweisungen vor jedem Eintrittspunkt. Wenn REM-Anweisungen aus Platzgründen gelöscht werden müssen, sollten die REM's vor den Eintrittspunkten als letzte beseitigt werden.

Problem Nr. 5: Erforderliche Initialisierungen

Alle Programme initialisieren sich selbst. Sie arbeiten deshalb auch, wenn im Hauptprogramm ihre Initialisierung vergessen wird. Dies ist ein wichtiges

Ziel für den Programmaufbau, denn auch unerfahrene Programmierer sollen unsere Unterprogramme verwenden können.

Unsere Lösung ist ein seltenes Beispiel eines APPLESOFT-Tricks (d.h. dieser Trick wird nicht notwendigerweise in anderen BASIC-Varianten funktionieren). Die Lösung beruht auf der Tatsache, daß mit dem Ausführen des RUN-Befehls alle arithmetischen Variablen Null gesetzt und alle Textvariablen als leer angenommen werden. Wann immer eine Initialisierung notwendig ist, wird ein Steuerzeichen abgefragt (z.B. in KLANG EINHEIT, Zeile 13000 bedeutet $WR\% = 0$ nicht initialisiert und $WR\% > 0$ bereits initialisiert). In EINGABE EINHEIT wird die Platzhalter-Variable YF\$ in Zeile 10000 initialisiert. Gleiches geschieht mit XS, dem Abstand zweier Bilder, in den Zeilen 15310 und 15410 im Unterprogramm BILD EINHEIT. Das automatische Festsetzen von DIM XC(10) in Zeile 15020 in BILD EINHEIT entspricht gerade noch den Ansichten des Programmierers für einen guten Programmierstil.

Problem Nr. 6: Gute Anpassung der Unterprogramme, Vermeiden von unerwünschten Nebeneffekten

Wie bereits erwähnt, ist dieses Problem stark von den persönlichen Ansichten des Programmierers abhängig. Um keine gegensätzlichen Meinungen darzustellen, wollen wir auf dieses Problem nicht weiter eingehen.

Ausgewählte Bemerkungen über die Programme

In Kapitel 1 werden zwei Programme in Maschinsprache benutzt. Das eine produziert Tonfrequenzen von festgelegter Dauer (KLANG), das andere produziert die Tonfrequenzen nur so lange, bis die nächste Taste gedrückt wird (ORGEL). Die Assemblerlistings sind im folgenden dargestellt:

```

33      *
34      * APPLESOFT ANGEPA SST
35      *
36      CLICK      EDU      $C030 ; LAUTSPR SCHALT
37      *
38      *
39      * KLANG -ERZEUGE KLANG MIT FESTGELEGTER TONDAUER
40      *
41      * EINGANG   : TONDAUER-L,H GESETZT
42      *           1: TONH SETZEN IN 'TONHTBL' (1/40)
43      *           2: TONH SETZEN
44      *
45      TONDAUER    DS      2
46      TONHOEHE    DS      1
47      *
48      * EINGANG 1: BENUTZE 'TONH' MIT AKTUELLEM
      TONHOEHEN-WERT
0320: AC 1F 03 49      KLANG1    LDY TONH
0323: B9 49 03 50              LDA  TONHTBL-1,Y
0326: B2 1F 03 51              STA  TONH
52      *
53      * EINGANG 2: 'TONH' SETZEN

```



```

0329: A0 00 54 KLANG2 LDY #0 ; INITIALISIERE 24-BIT
      'ZAEHLER'
032B: EE 1D 03 55 INC TONDAUER
032E: EE 1E 03 56 INC TONDAUER+1
      57 *
0331: AE 1F 03 58 NXTCLICK LDX TONH ;RESTORE TONH ZAEHLEN
0334: AD 30 C0 59 LDA CLICK ;'CLICK' LAUTSPR
      60 *
0337: 88 61 COUNTDOWN DEY ;24-BIT-ZAEHLER
      (Y, TONDAUER-L,H)
0338: D0 0A 62 BNE NICHT ERLEDIGT
033A: CE 1D 03 63 DEC TONDAUER
033D: D0 05 64 BNE NICHT ERLEDIGT
033F: CE 1E 03 65 DEC TONDAUER+1
0342: F0 05 66 BEQ ERLEDIGT
      67 *
0344: CA 68 N. ERL. DEX 'UEBERPRUEFE OB NAECHSTES CLICK'
0345: F0 EA 69 BEQ NXTCLICK
0347: D0 EE 70 BNE COUNTDOWN
      71 *
0349: 60 72 ERLEDIGT RTS
      73 *
      74 * TONHTBL - TONHOEHEN WERTE
034A: FF F2 E4 75 TONHTBL HEX FFF2E4D7CBC0B5AB ;1/8
0352: A1 98 8F 76 HEX A198F877F78716B ;9/16
035A: 65 5F 5A 77 HEX 655F5A55504B4743 ;17/24
0362: 3F 3B 38 78 HEX 3F3B3835322F2C2A ;25/32
036A: 2B 25 23 79 HEX 2B252321201E1C1A ;33/40
      81 *
      82 * APPLESOFT ANGEFASST
      83 *
      84 TASTE EQU $C000
      85 *
      86 *
      87 * ORGEL - ERZEUGE TONHOEHE BIS NEUE TASTE
      GEDRUECKT WIRD
      88 *
      89 * EINGANG: TONH SETZE IN 'TONHTBL' (1/40)
      90 *
0372: AC 1F 03 91 ORGEL LDY TONH
0375: B9 49 03 92 LDA TONHTBL-1,Y
0378: 8D 1F 03 93 STA TONH
037B: AD 00 C0 94 ORGCLICK LDA TASTE ;UEBERPRUEFE TASTATUR
037E: 30 0E 95 BMI ORGERL ;-) TASTE GEDRUECKT
0380: AE 1F 03 96 LDX TONH ; RESTORE TONHOEHEN ZAEHLER
0383: AD 30 C0 97 LDA CLICK ;'CLICK' LAUTSPR
      98 *
      99 * DIE NAECHSTEN BEIDEN ANWEISUNGEN
      WERDEN GEMACHT,
100 * UM DIE INNERE SCHLEIFE ZEITLICH
101 * DER VORHERIGEN PROZEDUR 'KLANG'
      ANZUPASSEN
102 *
103 * DIES FOLGT AUS DEN TONH-WERTEN,
104 * DIE IN JEDER PROZEDUR AEHNLICHE
      TONHOEHEN BEWIRKEN
105 *
0386: 88 106 ORGCOUNT DEY

```

```

0387:  D0 00      107          BNE  ORGNOP
                                EQU  * ; ENDE VON 'VERLUST'-ZEIT
0389:  CA          108  ORGNOP
                                DEX  ; UEBERPRUEFE OB NAECHSTES CLICK
038A:  F0 EF      109          BEQ  ORGCLICK
038C:  D0 F8      110          BNE  ORGCOUNT
                                112  *
038E:  60          113  ORGERL  RTS

```

Beachten Sie die zwei Eingänge von KLANG. In KLANG1 wird für die Tonhöhe (TONH) ein Wert in TONHTBL (Tonhöhentabelle) gesucht. In KLANG2 wird für TONH der aktuelle Wert benutzt. Die Töne entstehen durch das Ansteuern des Lautsprechers mit einer internen Frequenz, die durch die Werte in TONH bestimmt wird. Der Zusammenhang der internen Frequenz mit dem entstehenden Klang hängt von der zeitlichen Abstimmung des Maschinenprogramms ab. Beachten Sie, daß in TONHTBL Raum für vierzig interne Frequenzen ist. Der 16-bit-Wert in TONDAUER bestimmt die Länge des Tones.

Das Programm ORGEL benutzt die gleichen Frequenzen aus TONHTBL und verbraucht zusätzliche Rechenzeit in einer internen Schleife. Damit wird die zeitliche Anpassung zu dem KLANG-Programm vorgenommen. Im Gegensatz zu TONDAUER in KLANG fährt das ORGEL-Programm so lange mit dem Produzieren des Tones fort, bis eine neue Taste gedrückt wird. Eine der Grenzen der APPLESOFT-Hardware besteht darin, daß wir nicht feststellen können, ob eine gedrückte Taste wieder gelöst wird. Deshalb muß das ORGEL-Programm auf einen neuen Tastendruck warten, um enden zu können.

Im LO-RES-Kapitel wird die Tastatur durch das Programm EINGABE LAENGE in verschiedene Bereiche aufgeteilt. Jedem Bereich entspricht ein anderer interner Parameter. Diese Technik kann auch auf eigene Programme des Lesers angewendet werden und erfordert lediglich eine einfache Erweiterung von EINGABE EINHEIT.

Die Vorteile einer Unterprogramm-Bibliothek werden an dem Beispiel von SPIRALE KLANG deutlich. Das Programm SPIRALE2 benötigt nur drei zusätzliche BASIC-Befehle, um SPIRALE KLANG hervorzubringen. (Auch die LO-RES-Titelbilder in den letzten zwei Kapiteln wurden mit sehr geringem Programmaufwand hergestellt).

Das Programm BILD EINHEIT wurde bereits in dem entsprechenden Kapitel ausführlich diskutiert. Zu erwähnen bleibt noch, daß die Ausführung durch die Zuhilfenahme von Maschinensprache erheblich beschleunigt werden kann. Doch dies zu behandeln entspricht nicht der Aufgabe dieses Buches. Maschinensprache wurde nur dann benutzt, wenn eine Lösung in BASIC nicht möglich war. Die Programmlänge und die Einlesezeit von Diskette können verkürzt werden, indem man nur die benötigten Bilder ins Listing aufnimmt.

Wir sind froh, daß für die Hi-RES-Grafik kommerzielle Software-Pakete erhältlich sind. Die APPLE-Hardware kann vieles leisten, aber mit APPLE-

SOFT verhält sich das anders. Schauen Sie in die APPLESOFT-Programmieranleitung für den Fall, daß numerische Variablen, Felder oder das Programm selbst die für die HI-RES-Grafik vorgesehenen Pufferspeicher einnehmen. Im Anhang L finden Sie die Zeiger (Pointer) der Nullseite. Mit diesen kann nachgeprüft werden, ob dieser Fall eingetreten ist.

In EINGABE EINHEIT gibt es eine Sperrvorrichtung für den Gebrauch der ESC-Taste, denn diese spezielle Taste kann dazu benutzt werden, um aus dem gerade laufenden Spiel auszusteigen. Um dies zu erreichen, müssen wir jedoch ESC RETURN drücken. Die 'ESC'-Schrift wird außerdem auf dem Bildschirm wiederholt. Damit wird das Problem des versehentlichen Betätigens von ESC beseitigt, da die ESC-Funktion nicht von Beginn an aktiviert ist.

Der ONERRGOTO-Befehl ist eigentlich nutzlos, außer beim Verbessern von Programmen. Die Fehler 0 bis 224 sind logische Fehler, die durch Umstrukturierung des Programms beseitigt werden können. Da in EINGABE EINHEIT der INPUT-Befehl nicht benutzt wird, ist die Fehlermeldung 254 nicht möglich. Die Fehlermeldung 255 mit CTRL-C ist eine hübsche Idee, wird aber nicht richtig eingesetzt. Die Anwendung von RESUME bewirkt die Fortsetzung des Programms mit dem gerade ausgeführten Befehl (d.h. RESUME und CTRL-C bewirken das erneute Ausführen des letzten Befehls, anstatt mit dem nächsten Befehl fortzufahren). Leider wird CTRL-C nur beachtet, wenn das Programm eine Eingabe erwartet. Wird die Taste anderweitig betätigt, kann sich dies sehr schlimm auswirken. Vielleicht kann die Fehlermeldung 255 zum Ausdruck einer Abschiedsmeldung „Auf Wiedersehen“ vor dem Absturz des Programms benutzt werden. Wenn eine hervorragende Lösung entdeckt wird, hat der Benutzer auch noch die RESET-Taste (bzw. CTRL-RESET) zur Verfügung.

Das Programm STORY ist ein Beispiel für ein einfaches Spiel mit einem farbigen Titelbild in LO-RES, einer Fehlerabfang-Prozedur für den Abbruch eines Wortes am Zeilenende und einem Storyaufbau mit Fragen und DATA-Anweisungen.

In BLOCKOUT versuchen wir, einige Begrenzungen in der APPLESOFT Textdarstellung zu überwinden. Die SCRN-Funktion, die in Kapitel 2 nicht erörtert wurde, wird in Zeile 8350 benutzt und garantiert das Wechseln der Farbe eines Blocks.

Die beiden Programme MATCH und KONZENTRATION arbeiten mit der Veränderung von Datenstrukturen und zeigen einige beeindruckende visuelle Effekte. Beachten Sie in der BILDER-Bibliothek die auf den Kopf gestellten Buchstaben A und T für das Titelbild von MATCH. Versuchen Sie als eine weitere Herausforderung KONZENTRATION mit einem Schwarz-Weiß-Fernseher zu spielen und die kleinen Veränderungen im Kartenbild zu erkennen.

STERNE ist ein anderes altbekanntes und beliebtes Spiel, das durch die Verwendung von LO-RES-Grafik und von Tönen viel schöner wurde. Der Effekt des wachsenden Balkens entstand rein zufällig.

SIMON erinnert an manche Kombinationsspiele, die über eine Million Möglichkeiten besitzen. Das Programm verwendet möglichst wenig maschinensprachliche Aufrufe und ESC-Folgen in PRINT-Befehlen, denn sie beeinträchtigen die Lesbarkeit des Programms. Eine bessere Lösung ist das Erweitern von Computersprachen durch zusätzliche Kommandos (z.B. HOME anstatt CALL-936). Deshalb ziehen wir es vor, mit PRINT einige Leerzeichen auszudrucken, anstatt eine maschinensprachliche Prozedur aufzurufen, um das Ende einer Zeile zu löschen.

Anhang C

Hinweise zu den Programm listings

Die folgenden Hinweise sind für Leser gedacht, die alle Programme dieses Buches per Hand eintippen wollen. Vielleicht helfen sie etwas beim Lesen und Eingeben der Listings.

1. Die Unterprogramm-Einheiten sollten ohne die anderen Programmteile sofort auf Diskette gespeichert werden. Auf diese Art können die Unterprogramme jederzeit mit eigenen Programmen verbunden werden, und das Unterprogramm muß nur einmal eingetippt werden.
2. Damit unsere Unterprogramme sich nicht mit den Programmen des Lesers überschneiden, haben wir hohe Zeilennummern benutzt. Eigene Listings sollten deshalb nicht über die Zeile 10000 hinausgehen, können aber ab Zeile 50000 fortgesetzt werden.
3. Variablenamen, die mit W, X, Y und Z beginnen, sind möglichst zu vermeiden. Sie werden bereits in unseren Unterprogrammen benutzt.
4. Die folgenden Hinweise sollen dazu dienen, eventuelle Unklarheiten beim Lesen der Listings zu beseitigen:
 - Der Buchstabe I wird nicht als Variablenname benutzt. Auch als Teil eines Variablennamens, wie z.B. in AI oder ZI, haben wir ihn nicht verwendet. Der Buchstabe I kann sehr leicht mit der Zahl 1 verwechselt werden.
 - Um Verwechslungen mit der Zahl 0 zu vermeiden, wird auch der Buchstabe O nie als Variablenname oder als Teil davon benutzt. Die Variable AO gibt es also nicht.
 - In unseren Listings gibt es Variablenamen wie A1, B0 oder C9.
5. Die Zeilennummern sowie leere Zeilen und REM-Zeilen trennen die einzelnen Programmteile und Erläuterungen voneinander ab.
6. Wenn der zulässige Speicherbereich überschritten wird, können die REM-Anweisungen gelöscht werden. Für spätere Änderungen oder Fragen ist es jedoch besser, die REM's stehen zu lassen. Löschen Sie zuerst die ein-

führenden REM-Befehle und später die REM-Anweisungen, die zusammen mit anderen Befehlen in einer Zeile stehen. Diese REM's erläutern direkt die Auswirkung des dazugehörigen BASIC-Befehls. Die einführenden REM's beschreiben den Zugang zu den Unterprogrammen oder wie Programmänderungen durchzuführen sind.

7. Weiterhin können bei Überschreiten des zulässigen Speicherbereiches nicht benutzte Teile von EINGABE EINHEIT und BILD EINHEIT gelöscht werden. Zum Beispiel benutzt das Spiel STERNE nur die Buchstaben S, T, E, R und N. Alle anderen Bilder in BILD EINHEIT können gelöscht werden.

Anhang D

Das Entwickeln von Programmen

Das Schlagwort „benutzerfreundliche Software“ wird heutzutage oft gebraucht. In dem Maße, in dem die Zahl der käuflich erwerblichen Computerprogramme zunimmt, wählen die Kunden kritischer aus dem bestehenden Angebot aus. Sie suchen nicht nur nach funktionsfähigen Programmen, sondern wollen auch leicht anwendbare Programme erwerben. Sie sind nicht zufrieden mit Programmen, bei denen der Text aus dem Bildschirmbereich verschwindet, erforderliche Antworten umständlich einzugeben oder Fragestellungen unklar sind.

Im Verlaufe dieses Buches haben wir viele Vorschläge gemacht und Vereinbarungen getroffen, die benutzerfreundlich sind. Die Eingabe-Prozeduren mit Fehlerbehandlungsschleifen und hilfreichen Fehlermeldungen sind Beispiele benutzerfreundlicher Programmierung. Die ESC-Vereinbarung zum Verlassen eines Programmes ist ein weiteres Beispiel.

Dieser Anhang faßt noch einmal alle bisher geäußerten Vorschläge zusammen und fügt weitere hinzu. Mit der folgenden Prüfliste kann der Leser sowohl seine eigenen, als auch kommerzielle Programme auf ihre Benutzerfreundlichkeit überprüfen.

Wünschenswerte Merkmale in Lehr- und Lernprogrammen

- Einführung und Information sind auf gleicher Ebene wie die Durchführung des Programms
- Verzweigungen, um Informationen übergehen zu können
- Verzweigungen für Anfänger bzw. Fortgeschrittene
- Schwierigkeit der Aufgabe ist den erforderlichen Lesefähigkeiten angepaßt.
- Informationen zum Beenden oder Unterbrechen werden deutlich dargestellt.
- Geordnete, übersichtliche Bildgestaltung
- Deutliche Angaben der einzelnen Größen
- Konsistente Eingabeform (entweder INPUT oder GET)
- Zahl der erlaubten Versuche durch Benutzer regelbar

- Dauer der Informationsdarstellung durch Benutzer regelbar
- Schwierigkeitsgrad durch Benutzer regelbar
- Reaktionen auf eine richtige Antwort sind stärker als auf eine falsche Antwort
- Hilfreiche und keine negativen Reaktionen
- Leicht zugängliche Hilfsinformationen
- Fehlerbehandlungsschleifen und hilfreiche Hinweise
- Häufiges Löschen des Bildschirms
- Konsistenter Gebrauch von Informations- und Programmabbruch-Vereinbarungen

Folgendes ist zu vermeiden:

- Abbrechen von Wörtern am Ende der Zeile
- Lesen und Ausdrucken am unteren Ende des Bildschirms
- Ständig wechselndes Bild
- Unpassendes Setzen von Leerzeichen
- Text, der aus dem Bildschirmbereich verschwindet
- Zu viele blinkende Textzeilen
- Zu ausgiebiger Gebrauch von Klangeffekten, speziell sich wiederholender Melodien

Auch folgende Fragestellungen sind zu beachten:

- Eignet sich das Problem für den Einsatz eines Computers oder läßt es sich auf andere Art besser lösen?
- Ist das zur Durchführung eines Spiels erforderliche Denken mit einer Förderung der Lernerfahrung verbunden?
- Ist die Benutzung des Programms einfach genug? Es ist nicht der Zweck eines Spiels, daß der Benutzer mühsam lernt, wie das Spiel durchgeführt wird.

Sachwortverzeichnis

APPLESOFT II Benutzer-Handbuch 48, 49,
113
ASCII 2, 38, 39, 42

Beep 1, 2
BELL 1, 2
BILD EINHEIT:
– Listing 31–35
– Löschen des Bildschirms 37
– Positionierung in Bildschirmmitte 35
– Positionierung nicht in Bildschirmmitte 36
– weitere Bilder 40–48
– Zeichen-Bibliothek 39
– Zeilennummerierung 38
Black-box-Programm 5
BLOCKOUT:
– Erläuterungen 78, 82–83
– Listing 78–82
BLUMEN 20
Buchstabenspiele 38

COLOR 17
Community Computer Center 75, 100
CTRL G 1

DREI-FARBEN-BAUM 44–45

EINGABE EINHEIT 57–59
– Dezimalzahlen Eingabe 65
– Einzelne Zeichen Eingabe 66
– ESC-Vereinbarung 60
– Integer-Zahlen Eingabe 65
– J/N 65–66
– Pause oder Tastendruck 66–67
– Tastendruck ohne Wiedergabe 68–69
– Universelle Eingabe 59–63
– Zusammenfassung 69–70

EINGABE FARBEN 2

EINGABE LAENGE 23

EINGABE RECHTECKE 23

EINGABE TOENE 3

Entwickeln von Programmen 123–124

ESC-Vereinbarung 60

FARBIGE LINIEN 21

FARBPUNKTE 18–19

Farbtabelle:

– HI-RES 50
– LO-RES 17

GITTER NETZ 54
GITTER NETZ2 54
GR 17

Hangman 78
HCOLOR 50
HGR 50, 51
HGR2, 50 51
HI-RES DEMO1 51
HI-RES Grafik 49–55
– farbiges Löschen 51
– Farbtabellen 50, 51
– HCOLOR 50
– HGR 50
– HGR2 50
– HPLOT 50
– Probleme 52–53
HI-RES UMRANDUNG 52
HLIN 17
HOME 19
HPLOT 50–51

KLANG EFFEKTE:
– Erläuterungen 12–13
– Listing 12
Klang-Effekte Zusammenfassung 14–15
KLANG EINHEIT 6–7
KONZENTRATION:
– Erläuterungen 93, 99–100
– Listing 93–98

LAENGE LINIEN 21
LO-RES-Grafik 16–29
– COLOR 17
– Farbkarte 17
– Ganzer Bildschirm 28
– GR 17
– HLIN 17
– Löschen 24–25
– PLOT 17
– Textfenster 17, 18
– VLIN 17

MATCH:
– Erläuterungen 84, 90–91
– Listing 84–90
MUSIK BOTSCHAFT 9

NEXDATA EINHEIT 5

ORGEL 11

Palindrom 10

PIANO 10

PLOT 17

RAND1 24

RAND2 24

RENUMBER/MERGE-Hilfsprogramm 109–110

Schaltvorgänge 1, 4

SIMON 104–107

SPEED 2

SPIRALE 25

SPIRALE KLANG 27

SPIRALE1 26

SPIRALE2 26

SPIRALE3 26

STERNE 100–103

STORY:

– Erläuterungen 72, 75–77

– Listing 72–75

TASTEN A/Z 9

TASTEN 1/8 7–8

TEXT 19

Unterprogramme:

– Einzelne Zeichen Eingabe 66

– J/N 65–66

– Pause oder Tastendruck 66–67

– Tastendruck ohne Wiedergabe 68–69

– Universelle Eingabe 59–63

– Zahleneingabe 63–65

Überschreiben 76

VLIN 17

VOLLE LO-RES GRAFIK 28

VOR/ZURUECK 10

WASCHEN 25

WUNSCHBRUNNEN 19–20

ZAEHLE TOENE 4

Zeichen-Bibliothek 39

Zufallszahlberechnung 18

ZWEI-FARBEN-BAUM 40–41

ZWEI SPIRALEN 27

Mathematicals

Mathematische Unterhaltungen und Spiele mit dem
programmierbaren Taschenrechner (AOS)
von Hans H. Gloistehn

Logik unterm Galgen
von Martin Gardner

Mathematische Tricks
von Martin Gardner

Mathematische Knobeleyen
von Martin Gardner

Mathematische Rätsel und Probleme
von Martin Gardner

Mathematisches Labyrinth
von Martin Gardner

Mathematische Leckerbissen
von Stanley C. Ogilvy

Unterhaltsame Geometrie
von Stanley C. Ogilvy

Mathematische Edelsteine
von Ross Honsberger

Mathematische Juwelen
von Ross Honsberger

Gitter — Reste — Würfel
von Ross Honsberger